68' MICRO JOURNAL

Australia	A $ 4.75	New Zealand	NZ $ 6.50
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30.-SEK

$2.95 USA

OS-9 Atari Amiga Mac S-50

6800 6809 68008 68000 68010 68020 68030

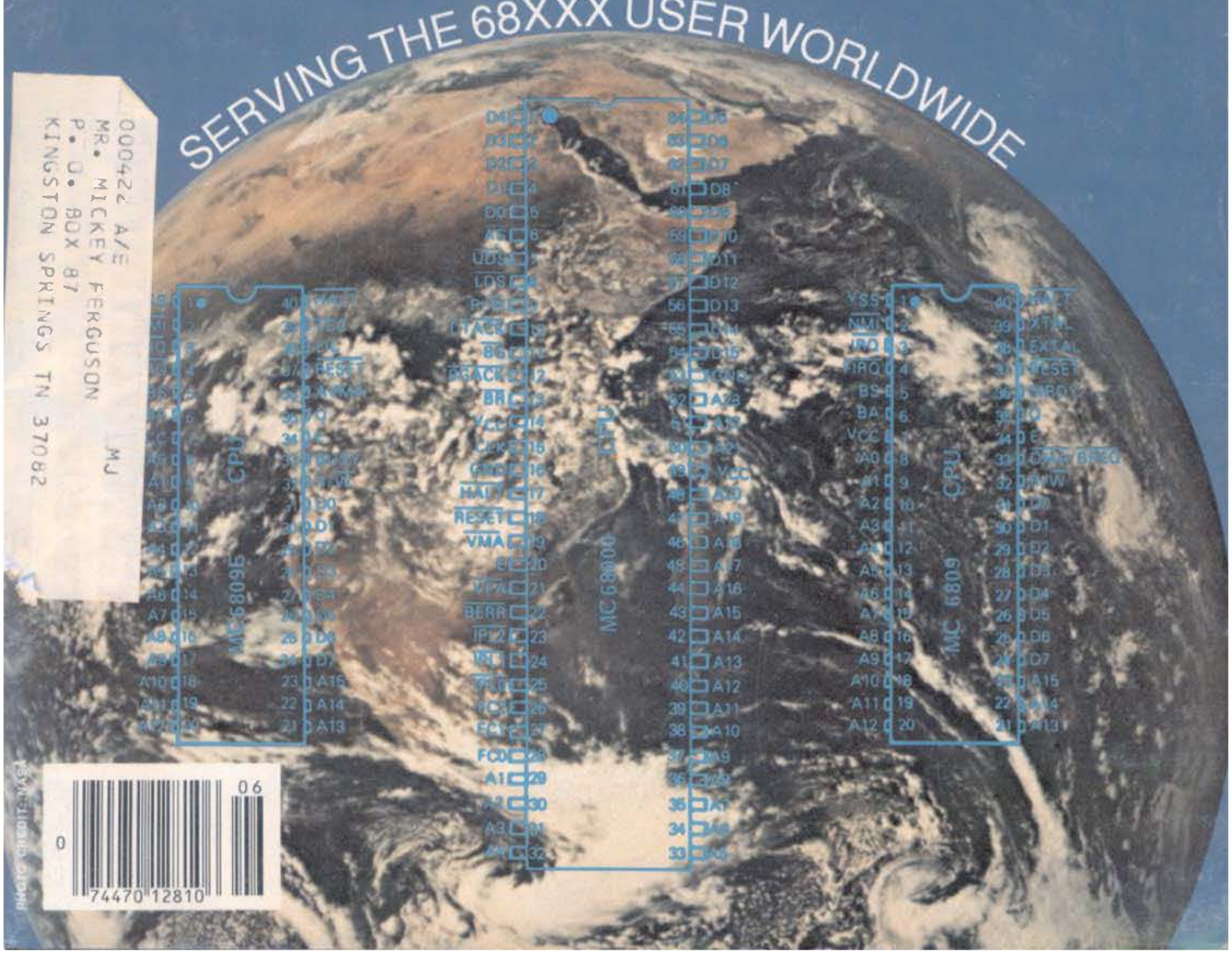
The Magazine for Motorola CPU Devices *For Over a Decade!*

OS-9 SK*DOS Atari Amiga FLEX Macintosh *A User Contributor Journal* And Lots More!

VOLUME X ISSUE VI ● Devoted to the 68XXX User ● June 1988

The Grandfather of "DeskTop Publishing™"

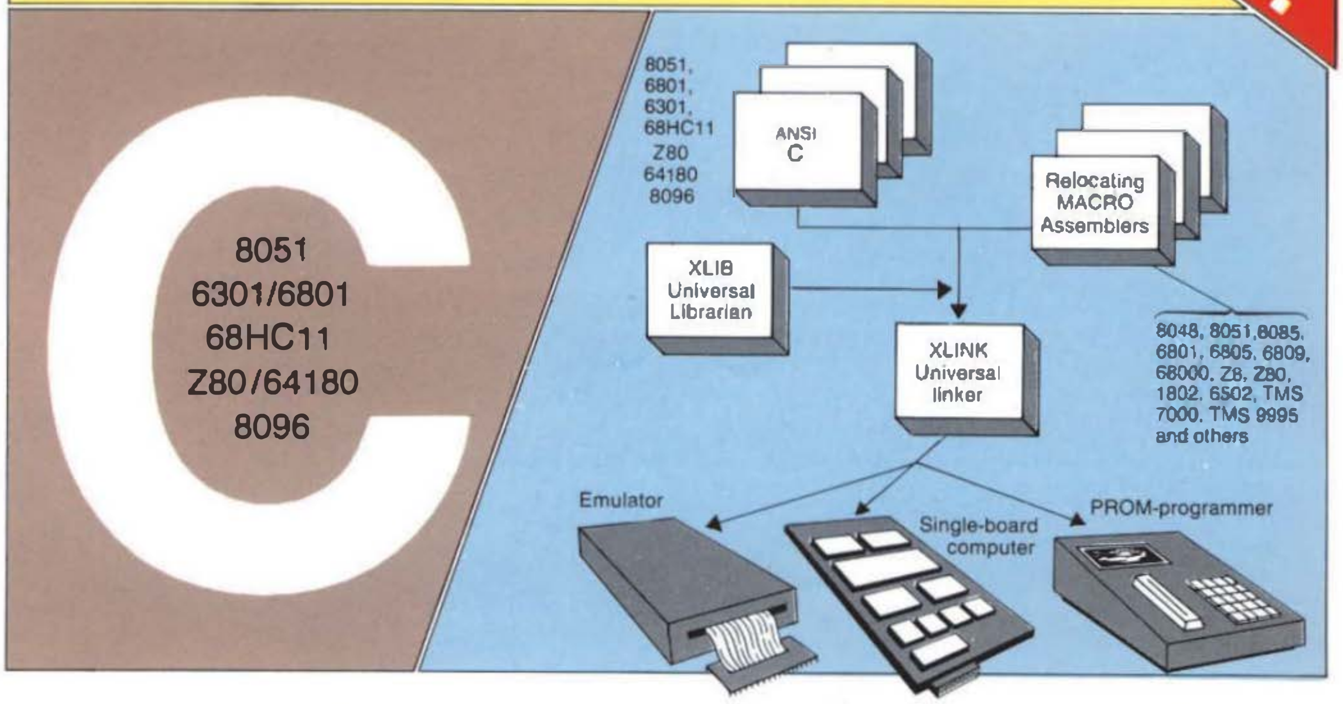
000422 A/E
MR. MICKEY FERGUSON MJ
P. O. BOX 87
KINGSTON SPRINGS TN 37082



06
0 74470 12810

ICC C CROSS COMPILER
FOR OS-9/68xxx BASED SYSTEMS

NEW!


8051
6301/6801
68HC11
Z80/64180
8096
8051, 6801, 6301, 68HC11 Z80 64180 8096
ANSI C
Relocating MACRO Assemblers
XLIB Universal Librarian
XLINK Universal linker
8048, 8051, 8085, 6801, 6805, 6809, 68000, Z8, Z80, 1802, 6502, TMS 7000, TMS 9995 and others
Emulator
Single-board computer
PROM-programmer

ANSI C

Full implementation of the proposed ANSI standard for C compilers. Includes the Kernighan & Ritchie standard plus improvements for micro-controller development.

Memory-based compiler

ICC is a fast one-pass compiler based on main memory storage. This has three major advantages:

- NO temporary files
- NO time-consuming assembly pass
- NO separate pre-processor stage

This combines into one single word: *SPEED*

PROM-able C

ICC makes it possible to put C programs into PROM, still using the full C language and all data types, including type definitions, long integers and statically initialized variables.

Built-in Type-Checking

ICC has a UNIX LINT-like type-checking option built-in into the compiler. This means that Pascal-like warnings are generated, e.g. when functions are mismatched or undeclared.

Various Options

- 8051 — single-chip
  — 64 K CODE + DATA
  — 64 K CODE + 64 K DATA
- 6301 and 6801
- Z80 and 64180

Full Package Development System

The ICC compiler package includes:

- C run-time library
- μ-Series Relocatable Macro Assembler
- XLINK Universal Linker
- XLIB Universal Librarian
- Floating-point support
- 150 page manual in three-ring binder

All this together give the micro-controller programmer a powerful Development System Software Environment.

For more information contact your local distributor:

Frank Hogg Laboratory
The Regency Tower
Suite 215
770 James Street
Syracuse, NY 13203
Phone: (315)474-7856
Telex: 646740

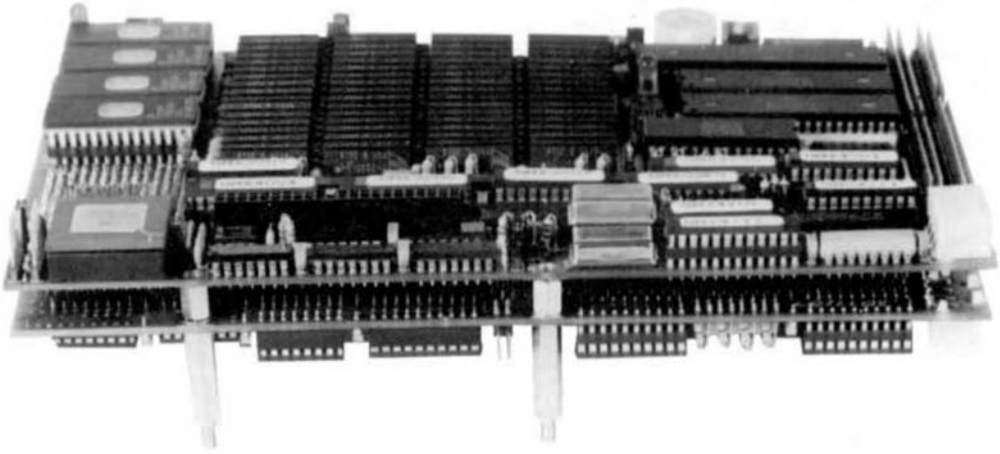
Elsoft AG
Zelweg 12
CH-5405 Baden-Dattwil
Switzerland
Phone: (056) 83 33 77
Telex: 828275

IAR SYSTEMS

OS-9/68xxx version distribution by:

Micromester Scandinavia AB
Box 1309, S-751 43 UPPSALA, SWEDEN.
Tel. inl.: + 4618 13 85 95. Telex: 76129

NOW THE GMX MICRO-20 HAS A TWIN — *Announcing The* GMX TWINGLE-20 68020 TWIN BOARD COMPUTER WITH MMU



All the features of the GMX Micro-20, PLUS —

- **2 Megabytes additional RAM — for a total of 4 Megabytes of RAM**
- **8 more Serial ports — for a total of 12, and expandable up to 44.**
- **MEMORY MANAGEMENT UNIT**

The GMX TWINGLE-20 consists of 2 boards. One of the boards is the same as the Micro-20, except for the 68020 processor which is on the MMU board. It uses the same I/O expansion interface, serial adapter boards, and mounting holes as the GMX Micro-20, making it easy to upgrade existing systems. Any of the currently available GMX Micro-20 I/O expansion boards can be used to provide additional I/O capability. Expansion possibilities include additional serial ports (up to 44 ports total), additional parallel ports, and local area networking of up to 255 GMX Micro-20s and/or TWINGLE-20s.

The MMU board contains the additional 2 Megabytes of RAM, 8 serial ports with 2 connectors for the SAB 4 port adaptor cards, and the MMU hardware. The MMU is a proprietary high-speed design that fully supports virtual memory. The system RAM normally operates with only 1 wait-state, regardless of processor speed. An additional wait-state is needed only when program flow crosses a 4K boundary. The MMU can be configured for any one of four different maps, ranging from 8 tasks with 8 megabytes of virtual address space each, to 64 tasks of 1 megabyte each. The MMU can be disabled for applications that do not use hardware memory management.

The TWINGLE-20 two board set can occupy the same space as a half-height 5.25" disk drive. It is available in 12.5, 16.67 or 20 MHz. versions, and with or without the 68881 FPC.

SPECIFICATIONS

Size: 8.8 × 5.75 × 1.4 inches.

Power Requirements: +5VDC @ 8.3A typical (20MHz. with 68881).
The TWINGLE-20 itself does not require a +12V supply. +12V supply requirements, if any, are determined by the serial adapter boards and any I/O expansion boards powered through the I/O Expansion Interface.

SOFTWARE INCLUDED:

An enhanced version of 020Bug with diagnostics for the MMU and the additional RAM and serial ports.

OPTIONAL SOFTWARE:

UniFLEX VM, Virtual Memory version of the UniFLEX operating system which includes all of the features of the GMX Micro-20 version, plus full MMU support.

The UniFLEX VM Operating System is a demand-paged, virtual memory operating system written in 68020 Assembler code for compactness and efficiency. Any UniFLEX system will run faster than a comparable system written in a higher level language. This is important in such areas as context switching, disk I/O, and system call handling. Other features include:

- Compact, efficient Kernel and modules allows handling more users more effectively than UNIX systems, using much less disk space.
- UNIX system V compatibility at the C source code level.
- C Compiler optimized in 68020 code (optional).
- Record locking for shared files.
- Users can share programs in memory.
- Modeled after UNIX systems, with similar commands.
- System accounting facilities.
- Sequential and random file access.
- Maximum record size limited only by the disk size.
- Multiple Level Directories.
- Up to 8 Megabytes of Virtual Memory per user.

All the optional languages and software that run under UniFLEX for the Micro-20 are also available for the TWINGLE-20.

OS-9 Users can take advantage of the additional RAM and serial ports on the TWINGLE-20. It does not presently support the MMU.

GMX inc 1337 W. 37th Place, Chicago, IL 60609 (312) 927-5510 — TWX 910-221-4055 — FAX (312) 927-7352

A Member of the CPI Family

68 Micro Journal

10 Years of Dedication to Motorola CPU Users

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

Publisher
Don Williams Sr.

Executive Editor
Larry Williams

Production Manager
Tom Williams

Office Manager
Joyce Williams

Subscriptions
Stacy Power

Contributing & Associate Editors

Ron Anderson — Dr. E.M. "Bud" Pass
Ron Voigts — Art Weller
Doug Lurie — Dr. Theo Elbert
Ed Law — & Hundreds More of Us

Contents

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.
"Over a Decade of Service"

"World Wide"

68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343
Phone (615) 842-4600 Telex 510 600-6630

68 Micro Journal is the *original* "DeskTop Publishing" product and has continously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

68 Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage paid ISSN 0194-5025 at Hixson, TN. and additional entries. Postmaster: send form 3597 to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year $24.50 USA, Canada & Mexico $34.00 a year. Others add $12.00 a year surface, $48.00 a year Airmail, USA funds. 2 years $42.50, 3 years $64.50 plus additional postage for each additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK*DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

P ase - do not format with spaces any text indents, char s, etc. (source listing o.k.). We will edit in all formatt ng. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.

Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word thereafter. No classifieds accepted by telephone.

OS-9 . . .
Making Beautiful Music in the Key of "C"!

When you need to orchestrate beautiful music on your VME system, look to Microware for just the right score. Our finely tuned OS-9 Operating System is truly the maestro's choice when the project requires C language development. Our superlative C Compiler—also available in an optimized 68020 version—produces fast, compact ROMable code for your most demanding applications. A powerful Assembler, Linker and Symbolic Debugger assists in target development. And our C Compiler is source compatible with UNIX applications and available in cross-compiler configurations for Sun, VAX and Hewlett-Packard environments.

OS-9 Keeps On Performing Even After the Fat Lady Sings!

Most operating systems hit a sour note when the project reaches completion. But not OS-9. Because of its modular design and UNIX-style architecture, your investment in OS-9 experience, tools and applications translates into a valuable resource for your company's future. OS-9 can be utilized again and again over your entire corporate product range.

An Accompaniment of Total Support

Microware is proudly setting the industry's standard for customer support. You'll find outstanding technical documentation that leaves nothing in doubt when it comes to real-world applications. A rigorous Quality Assurance program guarantees customer satisfaction by identifying trouble spots before they become customer problems. And with our Customer Hotline, you are only a telephone call away from courteous and concise information. So join the growing legions of Microware "C" aficionados. Call us today and find out how you can create inspiring harmonies on your system.

microware®

MICROWARE SYSTEMS CORPORATION

1900 N.W. 114th Street
Des Moines, IA 50322
Phone 515-224-1929

Western Regional Office
4401 Great America Parkway
Santa Clara, CA 95054
Phone 408-980-0201

Microware is a registered trademark of Microware Systems Corporation.
OS-9/68000 is a trademark of Microware. VAX is a trademark of DEC.
UNIX is a trademark of AT&T

MUSTANG -020 Super SBC ™

DATA-COMP Proudly Presents the First Under $4300 "SUPER MICRO" See other advertising (backcover) for economy system (68008) - under $2400 complete!

™ CPI
Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

The MUSTANG-020 68020 SBC provides a powerful, compact, 32 bit computer system featuring the "state of the art" Motorola 68020 "super" micro-processor. It comes standard with 2 megabyte of high-speed SIP dynamic RAM, serial and parallel ports, floppy disk controller, a SASI hard disk interface for intelligent hard disk controllers and a battery backed-up time-of-day clock. Provisions are made for the super powerful Motorola MC68881 floating point math co-processor, for heavy math and number crunching applications. An optional network interface uses one serial (four (4) standard, expandable to 20) as a 125/bit per second network channel. Supports as many as 32 nodes.

The MUSTANG-020 is ideally suited to a wide variety of applications. It provides a cost effective alternative to the other MC68020 systems now available. It is an excellent introductory tool to the world of hi-power, hi-speed new generation "super micros". In practical applications it has numerous applications, ranging from scientific to education. It is already being used by government agencies, labs, universities, business and practically every other critical applications center, worldwide, where true multi-user, multi-tasking needs exist. The MUSTANG-020 is UNIX C level V compatible. Where low cost and power is a must, the MUSTANG-020 is the answer, as many have discovered. Proving that price is not the standard for quality!

As a software development station, a general purpose scientific or small to medium business computer, or a super efficient real-time controller in process control, the MUSTANG-020 is the cost effective choice. With the optional MC68881 floating point math co-processor installed, it has the capability of systems costing many times over it's total acquisition cost.

With the DATA-COMP "total package", consisting of a heavy duty metal cabinet, switching power supply with rf/line by-passing, 5 inch DS/DD 80 track floppy, Xebec hard disk controller, 25 megabyte winchester hard disk, four serial RS-232 ports and a UNIX C level V compatible multi-tasking, multi-user operating system, the price is under $4300, w/12.5 megahertz system clock (limited time offer). Most all popular high level languages are available at very reasonable cost. The system is expandable to 32serial ports, at a cost of less than $65 per port, in multiples of 8 port expansion options.

The SBC fully populated, quality tested, with 4 serial ports pre-wired and board mounted is available for less that $2500. Quantity discounts are available for OEM and special applications, in quantity. All that is required to bring to complete "system" standards is a cabinet, power supply, disks and operating system. All these are available as separate items from DATA-COMP.

Available 12.5 - 25 Mhz systems, call for special prices

A special version of the Motorola 020-BUG is installed on each board. 020-BUG is a ROM based debugger package with facilities for downloading and executing user programs from a host system. It includes commands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassemble and numerous system diagnostics. Various 020-BUG system routines, such as I/O handlers are available for user programs.

Normal system speed is 3-4.5 MIPS, with burst up to 10 MIPS, at 16.6 megahertz. Intelligent I/O available for some operating systems.

Hands-on "actual experience sessions", before you buy, are available from DATA-COMP. Call or write for additional information or pricing.

Data-Comp Division
CPI — A Decade of Quality Service* — *Systems World-Wide*
Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

Mustang-020 Mustang-08 Benchmarks

	32 bit Integer	Register Long
IBM AT 7300 Xenix Sys 3	9.7	
AT&T 7300 UNIX PC 68010	7.2	4.3
DEC VAX 11/780 UNIX Berkley 4.2	3.6	3.2
DEC VAX 11/750 " "	5.1	3.2
68008 OS-9 68K 8 Mhz	18.0	9.0
68000 OS-9 68K 10 Mhz	6.5	4.0
MUSTANG-08 68008 OS-9 68K 10 Mhz	*9.8*	*6.3*
MUSTANG-020 68020 OS-9 68K 16 Mhz	*2.2*	*0.88*
MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz	*1.8*	*1.22*

```
Main()
            {
                register long i;
                for (i=0; i < 999999; ++i);
            }
```

Estimated MIPS - MUSTANG-020 — 4.5 MIPS,
Burst to 8 - 10 MIPS: Motorola Specs

Mustang Specifications

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP RAM (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
   Tered 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calendar w/battery backup
controller for 2, 5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48-96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single step capability

Don't be mislead! ONLY Data-Comp delivers the Super MUSTANG-020

OS-9

OS-9 Professional Ver	$850.00
*Includes C Compiler	
Basic09	300.00
C Compiler	500.00
68000 Disassembler (w/source add: $100.00)	100.00
Fortran 77	750.00
Microware Pascal	500.00
Omegasoft Pascal	900.00
Stylo-Graph	495.00
Stylo-Spell	195.00
Stylo-Merge	175.00
Stylo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUST w/C source	79.95
PAT/JUST Combo	249.50
Sculptor+ (see below)	995.00
COM	125.00

UniFLEX

UniFLEX (68020 ver)	$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/PreCompiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Sculptor+ (see below)	995.00

Standard MUSTANG-020™ shipped 12.5 Mhz.

Add for 16.6 Mhz 68020	375.00
Add for 16.6 Mhz 68881	375.00
Add for 20 Mhz 68020/RAM	750.00
16 Port exp. RS-232	335.00
Requires 1 or 2 Adapter Cards below RS232 Adapter	165.00

Each card supports 4 additional ser. ports
(total of 36 serial ports supported)

60 line Parallel I/O card	398.00

Uses 3 68230 Interface/Timer chips,
6 groups of 8 lines each, separate buffer
direction control for each group.

Prototype Board	75.00

areas for both dip and PGA devices & a
pre-wired memory area up to 512K DRAM.

SBC-AN	475.00

Interface between the system and
ARCNET modified token-passing LAN. fiber optics optional - call.

LAN software drivers	120.00
Expansion for Motorola I/O Channel Modules	$195.00

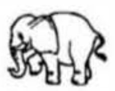
Special for complete MUSTANG-020™ system buyers - Sculptor+ $695.00. SAVE $300.00
Software Discounts

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

The PRO!

These hi-speed 68020 systems are presently working at NASA, Atomic Energy Commission, Government Agencies as well as Universities, Business, Labs, and other Critical Applications Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatability and low cost is a must.

Only the "PRO" Version of OS-9 Supported!



This is *HEAVY DUTY* Country!

For a limited time we will offer a $400 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation. Call for more information

Price List:

Mustang-020 SBC		$2490.00
Cabinet w/switching PS		$299.95
5"-80 track floppy DS/DD		$269.95
Floppy Cable		$39.95
OS-9 68K Professional Version		$850.00
C Compiler ($500 Value)		N/C
Winchester Cable		$39.95
Winchester Drive 25 Mbyte		$895.00
Hard Disk Controller		$395.00
Shipping USA UPS		$20.00
UniFLEX	Less	$100.00
MC68881 f/p math processor	Add	$275.00
16.67 Mhz MC68020		$375.00
16.67 Mhz MC68881		$375.00
20 Mhz MC68020 Sys		$750.00
Note all 68881 chips work with 20 Mhz Sys		
	Total:	$5299.80

Save $1000.00
Complete
25 Mbyte HD System
$4299.80
85Mbyte HD System
$5748.80

Note: Only Professional OS-9 Now Available (68020 Version)
Includes ($500) C Compiler - 68020 & 68881 Supported -
For UPGRADES Write or Call for Professional OS-9 Upgrade Kit

Data-Comp Division
CPI A Decade of Quality Service*
Systems World-Wide
Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

PAT - JUST

PAT

With 'C' Source

$229.00

All OS-9 68XXX Systems

service

For Over a Decade!
1,200 Titles +

A Division of
CPI

S.E. Media
5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

[illegible]10-68020

PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved **PIE**, this is for **YOU!** **All PIE features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K $229.00

COMBO ——— PAT/JUST

Special $249.00

JUST

JUST from S. E. MEDIA - - Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source $79.95

The C Programmers Reference Source. Always Right On Target!

C User Notes

A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

INTRODUCTION

This chapter presents still another version of a routine to scan command lines and two utility programs which are heavily dependent upon this scanner. They were all written by John Weald and placed into the public domain for the rest of us to use. The scanner is a version of getopt, a UNIX function, and the utility programs are versions of cut and paste, useful UNIX commands for text processing.

GENERAL

Command line scanners are not difficult to write; however, many have been written without thought as to how they are to be used. Some seem to have been designed to be difficult or obscure in their use, for no apparent reason.

Ideally, the order of options with respect to other arguments, such as file names, should logically be irrelevant in most programs. Thus, the following sequences would then generate equivalent results:

```
prog -a -b -c file1 file2 file3
prog file1 file2 file3 -a -b -c
prog -a file1 -b file2 -c file3
```

Such a capability is appreciated by many who use computers heavily and do not wish to consult manuals concerning the order of options. If order dependencies are present in a given program, they should be documented in a usage message (all programs have good usage messages, don't they?) or in a manual.

Many commonly-used programs on popular operating systems have such dependencies, and it is left to the programmer to remember the requirements and peculiarities of each program. The really treacherous programs accept the options without error but do not act on them as expected.

Microsoft C version 5.0 command-line arguments are scanned in a left to right manner. Thus, in the following example,

```
cl prog1.c -Dmsc -Am
```

the -Dmsc and -Am options would not be processed until after prog1.c is processed, potentially leading to a program being silently compiled with incorrect options.

The getopt scanner below provides the same capability as one by a similar name defined under UNIX. Unfortunately, it is not an ideal scanner in that it requires all options to be coded before other arguments, unless the programmer makes the extra effort to call it at appropriate times. Remember to provide an indicative usage message in such cases.

GETOPT

Getopt is declared with the following formal parameters:

```
int getopt (argc, argv, optstring)
int argc;
char **argv, *optstring;
```

It has the following external references:

```
extern char *optarg;
extern int optind, opterr;
```

Getopt returns the next option letter in argv (starting from argv[1]) which matches a letter in optstring, which is a string of recognized option letters; if a letter is followed by a colon, the option is expected to have an argument that may or may not be separated from it. Optarg is set to point to the start of the option argument on return.

Getopt places in optind the argv index of the next argument to be processed. Optind is initialized to 1 before the first call to the getopt function.

Options can be any ASCII characters except colon, question mark, or null. When all options have been processed, getopt returns EOF. The special option — may be used to delimit the end of the options; EOF will be returned, and — will be ignored.

Getopt normally outputs an error message to stderr and returns a question mark when it encounters an option letter not included in optstring. It is impossible to distinguish between a ? used as a legal option, and the character that getopt returns when it encounters an invalid option character in the input. This error message may be suppressed by setting opterr to zero.

```
/*
 *  This is a clone of the UNIX getopt() function.
 *
 * John Weald
 */

#include <stdio.h>

/*
 * Index into error array.
 */
#define BAD_OPT 0 /* option letter not in optstr */
#define MIS_ARG 1 /* option must have an argument */

char *optarg;
char *p;
char errors[2][30];
int opterr = 1; /* If true print error message */
int optind = 1; /* argv[0] is program name */

/*
 * The basic data structures are optind
 * and the pointer p.  optind keeps track
 * of the next index into argv to parse
 * arguments.  p is used to walk along the
 * argv items looking for option letters or
 * arguments, when it is NULL the next argv
 * must be used.  p is always left pointing
 * to the previous option or NULL.
 * Consider the three equivalent argv's:
 *        1        2          3
 *        -a      -b        eric
 *             -ab    eric
 *        -aberic
 */

getopt(argc, argv, optstr)
int argc;
char *argv[];
char *optstr; /* The list of valid options */
{
    extern err();     /* forward reference */

    /*
     * parsed all the options in this argv[]?
     */
    if (!p || !*++p)
    {
        if (optind == argc)
            return(EOF);
        p = argv[optind];

        /*  a '-' by itself is not an option */
        if (*p++ != '-' || !*p)
            return(EOF);

        /* '--' marks end of the option list */
        if (*p == '-')
        {
            optind++;
            return(EOF);
        }
    }

    optind++;
    /*
     * Look for a valid option
     */
    while (*p != *optstr)
    {
        if (!*optstr)
        {
            /* Reached end of optstr */
            err(argv[0], BAD_OPT, *p);
            return((int)'?');
        }
        if (*++optstr == ':')
            optstr++;
    }

    /* If needs no argument we are done. */
    if (*(optstr + 1) != ':')
        return((int)*optstr);

    /*
     * If there are more characters,
     * they must be the argument.
     */
    if (*++p)
    {
        optarg = p;
        p = NULL;
        return((int)*optstr);
    }

    /*
     * It needs an argument, but have none
     */
    if (optind == argc)
    {
        err(argv[0], MIS_ARG, *optstr);
        p = NULL;
        return((int)'?');
    }

    /*
     * Must be in next argv.
     */
    optarg = argv[optind++];
    p = NULL;
    return((int)*optstr);
}


err(a0, e, c)
char *a0; /* argv[0]: the program name */
int e;
char c;
{
    if (!*errors[0])
    {
        strcpy(errors[0],
            "%s: illegal option - %c\n");
        strcpy(errors[1],
            "%s: option needs argument - %c\n");
    }
    if (opterr)
        fprintf(stderr, errors[e], a0, c);
}
```

CUT

The cut utility program allows the user to extract columnar fields from one or more files. It concatenates the fields together horizontally separating them by spaces or delimiters, and appends a new-line character after each line.

It is invoked in one of the following manners:

```
cut -clist [file-list]
cut -flist [-dchar] [-s] [file-list]
```

When used in the first manner, the list provides a set of character positions defining the columns.

When used in the second manner, the list provides a set of fields delimited by single characters.

The options are interpreted as follows:

-clist The list following -c specifies character positions (base one). For example, -c1-8,17-24,32 specifies character positions 1 thru 8, 17 thru 24, and 32. Either the -c or the -f option must be specified.

-flist The list following -f specifies field numbers (base one). These fields are separated by character delimiters, as specified by -d, or the tab character by default. For example, -c1-8,17-24,32 specifies field numbers 1 thru 8, 17 thru 24, and 32. Either the -c or the -f option must be specified.

-dchar The character following -d is the field delimiter character for the -f option. The default delimiter character is the tab character.

-s Suppresses lines with no delimiter characters with the -f option.

```
/*
 *  This is a clone of the UNIX cut utility,
 *   except that the list of numbers
 *   does not have to be ascending.
 *
 * John Weald
 */

#include <stdio.h>
#include <ctype.h>

#define MAXLINE 1024 /* The max. length of a line */

main(argc, argv)
int argc;
char *argv[];
{
    extern char *optarg;
    extern int optind;

    FILE *fp; /* All the input files or stdin */
    char buf[MAXLINE]; /* The input buffer */
    char fs; /* The field separator */
    int c; /* The command line option */
    int cflag; /* True if -c on command line */
    int err; /* True if error in command line */
    int fflag; /* True if -f on command line */
    int suppress; /* Suppress lines with no delimiter */
    /* The field markers. True if this field */
    /* is to be cut, False otherwise */
    static int fields[MAXLINE];

    for (err = 0; err < MAXLINE; err++)
    {
        fields[err] = 0;
        buf[err] = NULL;
    }

    err = fflag = cflag = suppress = 0;
    fs = '\t';
    while ((c = getopt(argc, argv, "f:d:c:s")) != EOF)
    {
        switch (c)
        {
        case 'f': /* By Field */
            list(fields, optarg);
            fflag++;
            if (cflag)
                err++;
            break;

        case 'c': /* By character */
            list(fields, optarg);
            /* Implied suppress */
            suppress++;
            cflag++;
            if (fflag)
                err++;
            break;

        case 'd': /* A new field separator */
            fs = *optarg;
            break;

        case 's': /* Suppress if no delimiter */
            suppress++;
            break;

        default:
            prusage();
        }
    }

    if (!cflag && !fflag)
    {
        fprintf(stderr,
            "cut: Must have one of -f or -c\n");
        err++;
    }
    if (err)
        prusage();


    /* Loop on all the files.  */
    do
    {
        if (optind == argc)
            fp = stdin;
        else
        {
            if (!(fp = fopen(argv[optind], "r")))
            {
                fprintf(stderr,
                    "cut: Failed to open file %s\n",
                argv[optind]);
                exit(1);
            }
        }

        /* Loop on all lines in the file.  */
        while (fgets(buf, 1024, fp))
            cut(buf, fields, fs, suppress, cflag);
        (void)fclose(fp);
    }
    while (++optind < argc);

    exit(0);
}


/*
 * Cut the line.
 * This handles both character and field cutting.
 * For characters f array gives character positions.
 * For fields it gives the field number.
 * It must be indexed by either the
 *  character number or the field number.
 */
cut(in, f, fs, sup, c_or_f)
register char *in; /* The input line */
```

```
    int f[]; /* The field cutting flags */
    char fs; /* The field seperator */
    int sup; /* Suppress lines with no-delimiter?    */
    int c_or_f; /* Cut by char. (true), or field (false)*/
    {
        char *instart; /* To print lines with no delimit-
ers */
        char obuf[MAXLINE]; /* Output buffer */
        register char *optr;
        register int fld; /* The field count */
        register int i; /* Character count */

        instart = in;
        optr = obuf;
        for (i = 0; i < MAXLINE; i++)
            obuf[i] = NULL;
        for (fld = 0, i = 0; i < MAXLINE; i++)
        {
            if (*in == '\n')
            {
                /* End of the line */

                *optr = '\0';
                /* Anything to cut? */
                if (optr != obuf)
                {
                    /* Get rid of trailing separator */
                    if (*(optr - 1) == fs)
                        *(optr - 1) = 0;
                    puts(obuf);
                    return;
                }
                if (!sup)
                    printf(instart);
                return;
            }

            if (f[c_or_f ? i : fld])
                *optr++ = *in;

            /* End of field? */
            if (*in++ == fs)
                fld++;
        }

        fprintf(stderr,
            "cut: Line too long, maximum length is %d\n",
            MAXLINE);
        exit(1);
    }


    /*
     * Parse the list argument. The format is:
     *     n,n
     *  where n is either a number or a range of
     *  numbers in the format
     *     m-l
     * m or l may be absent, indicating the start
     *  or end of the lines respectivly.
     * Numbers must be in increasing order for m-l
     *  format, but not for n,n.
     * Field numbers start at 1, but index into
     *  fields array starts at 0.
     *
     */
    list(f, l)
    int f[]; /* The fields */
    char *l; /* The list */
    {
        int i;
        int high; /* the low and high numbers in a m-l
pair*/
        int low;
        int range; /* True if m-l format */

        low = 1;
        high = range = i = 0;

        while (1)
        {
            switch (*l)
            {
            case '\0':
                /* Is it m-<nothing>EOL? */
                if (range)
                {
                    /* Select rest of fields */
                    for (i = low - 1; i < high; i++)
                        f[i] = 1;
                }
                else
                    f[low-1] = 1;
                return;

            case ',':
                l++;
                if (!range)
                {
                    f[low-1] = 1;
                    range = 0;
                    low = 1;
                }
                else
                {
                    if (isdigit((int)*l))
                    {
                        low = atoi(l);
                        /* Skip the digits */
                        while (isdigit((int) *l))
                            l++;
                    }
                    range = 0;
                }
                break;

            case '-':
                l++;
                range++;
                /* Is it m-<nothing> */
                if (isdigit((int)*l))
                {
                    high = atoi(l);
                    /* Skip the digits */
                    while (isdigit((int) *l))
                        l++;
                }
                else
                    high = MAXLINE;

                /* Is the range the correct way around? */
                if (low > high)
                {
                    fprintf(stderr,
                        "cut: Bad c/f list: %d > %d\n",
                        low, high);
                    exit(1);
                }

                /* Set the field flags for the range */
                for (i = low - 1; i < high; i++)
                    f[i] = 1;
                break;

            default:
                /* either a number or an error */
                if (!isdigit((int)*l))
                {
                    fprintf(stderr,
                        "cut: Bad c/f list at %s\n", l);
                    exit(1);
                }

                if (!(low = atoi(l)))
                {
                    fprintf(stderr,
                        "cut: Fields start at 1 not 0\n");
                    exit(1);
                }
                /* Skip the digits */
                while (isdigit((int) *l))
                    l++;
                break;
            }
        }
    }


    prusage()
    {
        fprintf(stderr, "Usage: cut [-d<delimiter>] [-
s]");
        fprintf(stderr, " -c<list>|-f<list> [file-
list]\n");
        exit(1);
    }
```

PASTE

The paste utility program allows the user to merge corresponding lines of several files or to merge subsequent lines of the same file.

It is invoked in one of the following manners:

paste [file-list]
paste -dlist [file-list]
paste -s [-dlist] [file-list]

In the first two manners, paste concatenates corresponding lines of the input files. It treats each file as columns of a table and concatenates them together horizontally.

In the last manner, paste combines subsequent lines of each input file.

In all cases, lines are separated by the tab character, or with characters from an optionally-specified list.

The options are interpreted as follows:

-dlist This option allows the specification of the field separation character list, which is a tab character by default. The list is used circularly. In parallel file merging (not with the -s option), the lines from the last file are always terminated with a new-line character, not from the list. The list may contain the special escape sequences: \n (new-line), \t (tab), \\ (backslash), and \0 (empty string, not a null character).

-s This option may be used to merge subsequent lines from the same file rather than corresponding ones from each input file. Regardless of the contents of or absence of a list, the very last character of the file is forced to be a new-line.

- This option may be used in place of any file name, to read from the standard input.

```
    /*
     *  This is a clone of the UNIX paste utility.
     *
     * John Weald
     */

    #include <stdio.h>

    #define MAXLINE 1024 /* Max. allowed line length */
    #define PLUSLINE 1025
    #define MAXFILES 12 /* Max. number of input files */

    main(argc, argv)
    int argc;
    char *argv[];
    {
        extern int optind;
        extern char *optarg;

        char conchars[MAXFILES]; /* The concatenation
characters */
        int c; /* For getopt() */
        int nconchars; /* The number of conchars[] */
        int serial; /* True if type paste "-s" */

        conchars[0] = '\t';

        nconchars = 1;
        serial = 0;

        while ((c = getopt(argc, argv, "sd:")) != EOF)
        {
            switch (c)
            {
            case 's': /* Concatenate the same file serially
*/
                serial++;
                break;

            case 'd': /* Use other than a single tab */
                nconchars = setconcat(conchars, optarg);
                break;

            default: /* Does not return */
                prusage();
            }
        }

        if (serial)
            spaste(&argv[optind], conchars, nconchars);
        else
            paste(&argv[optind], conchars, nconchars);
        exit(0);
    }


    /*
     * paste()
     *
     * Do the actual paste.
     */
    paste(files, con, ncons)
    char *files[]; /* Null terminated list of input files */
    char con[]; /* The concatenation characters */
    char ncons; /* The number of above */
    {
        FILE *fps[MAXFILES]; /* One for each open file */
        char c; /* The current concatenation char */
        char ibuf[PLUSLINE]; /* The input buffer */
        char obuf[MAXLINE]; /* The output buffer */
        int allfiles;
        int f; /* Number of files opened */
        int i;
        int inc; /* True if concatenation character == '\0'
*/
        int ocount; /* Output buffer character count */
        register char *iptr;
        register char *optr;

        iptr = ibuf;
        optr = obuf;

        /*
```

```
         * Open all the input files, any filename of '-'
means
         * the standard input. No file name means standard
input.
         */
        for (f = 0; files[f]; f++)
        {
            if (*files[f] == '-')
                fps[f] = stdin;
            else
            {
                if (!(fps[f] = fopen(files[f], "r")))
                {
                    fprintf(stderr,
                        "paste: Failed to open file %s\n",
files[f]);
                    exit(1);
                }
            }
            if (f >= MAXFILES)
            {
                fprintf(stderr,
                    "paste: Too many files. Maximum
allowed is %d\n",
                    MAXFILES);
                exit(1);
            }
        }
        if (!files[0])
        {
            fps[0] = stdin;
            f++;
        }

        /* Read all lines until no more lines in any file.
*/
        allfiles = f;
        while (f)
        {
            optr = obuf;
            ocount = 0;
            /* Join lines from all files.
             * The concatenation character may be '\0'
which
             * means no character. The variable inc is an
indication
             * of the concatenation character being '\0',
we need to
             * if there is a concatenation character to
move up the
             * output buffer.
             *
             * The concatenation characters are used in a
round robin
             * list.
             */

            for (inc = 0, i = 0; i < allfiles; i++)
            {
                iptr = ibuf;
                optr += inc;
                /* To save repeated evaluation */
                inc = !(c = con[i % ncons]);

                if (!fps[i])
                {
                    /* No more lines in this file. */
                    *optr = c;
                    continue;
                }
                if (!fgets(ibuf, sizeof(ibuf), fps[i]))
                {
                    /* Reached EOF - finished with the
file */

                    (void)fclose(fps[i]);
                    fps[i] = NULL;
                    *optr = c;
                    f--;
                    continue;
                }
```

```
                /*
                 * Replace newline with the concatenation
character.
                 * There is no need to look for end-of-
string since
                 * we know that
                 * a) if ibuf is full to the max, then we
will
                 *    overflow obuf before we hit the end of
ibuf.
                 * b) if ibuf is not full, then it must
contain a
                 *    a newline character, but may or may
not
                 *    fit into obuf.
                 */
                for ( ; *iptr != '\n'; ocount++)
                {
                    /* Need space for trailing null */
                    if (ocount >= sizeof(obuf) - 1)
                    {
                        fprintf(stderr,
                            "paste: Output line too long,
maximum is %d.\n",
                            MAXLINE);
                        exit(1);
                    }
                    *optr++ = *iptr++;
                }
                *optr = c;

            }
            if (f)
            {
                *optr = 0;
                puts(obuf);
            }
        }
    }


    /*
     * setconcat()
     *
     * Parse the concatenation characters and place them in
the array c.
     * Return the number of concatenation characters.
     *
     * Specials are:
     *     \n     - Newline
     *     \t     - Tab (default)
     *     \     - Backslash
     *     \0     - No concatenation character
     */
    static int
    setconcat(c, in)
    char *c;
    char *in;
    {
        int i; /* The number seen so far */

        for (i = 0; *in; in++, c++, i++)
        {
            if (i > MAXFILES)
            {
                fprintf(stderr,
        "paste: Too many concatenation characters, more than
%d\n",
                    MAXFILES);
                exit(1);
            }
            if (*in != '\\')
            {
                *c = *in;
                continue;
            }
            switch (*++in)
            {
            case 'n':
                *c = '\n';
                break;
            case 't':
                *c = '\t';
                break;
            case '0':
                *c = 0;
                break;
            default: /* Includes '\\' */
                *c = *in;
                break;
            }
        }
        return(1);
    }


    static
    prusage()
    {
        fprintf(stderr, "Usage: paste [-s] [-d<list>]
files\n");
        exit(1);
    }


    spaste(files, c, n)
    char *files[]; /* Null terminated list of input files */
    char c[]; /* The concatentaion characters */
    int n; /* The number of above */
    {
        FILE *fp;
        int i;

        if (!files[0])
        {
            spfile(stdin, c, n);
            return;
        }
        for (i = 0; files[i]; i++)
        {
            if (*files[i] == '-')
                fp = stdin;
            else
            {
                if (!(fp = fopen(files[i], "r")))
                {
                    fprintf(stderr,
                        "Failed to open file %s\n",
files[i]);
                    exit(1);
                }
                spfile(fp, c, n);
                (void)fclose(fp);
            }
        }
    }


    /*
     * Do the actual paste of a stream.
     *
     * The method here is to read in the stream and replace
     * newline characters with concatentaion characters.
     * Output occurs after each chuck is parsed, or if
     * the concatenation character is the null separator.
     *
     * The stream is read in BUFSIZ chunks using fread.
     * The input buffer is one character larger than read,
     * so that it can be null terminated.
     *
     * When we read in each chunk we must check if it
     * needs to be joined to the previous one.
     */
    static
    spfile(fp, con, ncons)
    FILE *fp; /* serially paste this stream */
    char con[]; /* The concatentaion characters */
    int ncons; /* The number of above */
    {
        char *pstart; /* The start of the string */
        char *ptr; /* Walks down the stream */
        char buf[BUFSIZ+1]; /* To ensure null termination */
        char last; /* The very last character looked at */
        int join; /* Join this chunk to the next? */
        int k; /* Index into concatenation character array
*/
        int n; /* Number of bytes read with fread() */

        join = k = 0;
        while ((n = fread(buf, sizeof(char),
            sizeof(buf) - 1, fp)))
        {
            if (join)
            {
                /* Join with last chunk */
                putchar(con[k]);
                k = (k + 1) % ncons;
                join = 0;
            }
            /* Join to next chunk? */
            if (buf[n-1] == '\n')
            {
                join++;
                /* Ignore the newline */
                n--;
            }

            /* ensure null terminated buffer */
            buf[n] = '\0';

            /* walk thru this chunk */
            /* replace newlines with next concat. char. */
            for (pstart = ptr = buf; *ptr; ptr++)
            {
                if (*ptr == '\n')
                {
                    *ptr = con[k];
                    if (!con[k])
                    {
                        fputs(pstart, stdout);
                        pstart = ptr + 1;
                    }
                    k = (k + 1) % ncons;
                }
            }
            fputs(pstart, stdout);
            last = *(ptr - 1);
        }

        /*
         * Check for the newline as concatenation char.
         */
        if (last != '\n')
            putchar('\n');
    }
```

EOF

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

Basically OS-9

Dedicated to the serious OS-9 user.
The fastest growing users group world-wide!
6809 - 68020

A Tutorial Series

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL

PARDON IF I INTERRUPT

I have to relate an incident that happened to me some years back. It was about 4 years ago. I had bought a modem. It was a birthday present to myself. I was very enthusiastic about getting one. Around that time modems were the thing to have. Everyone had one. I knew many computerists whose only existence was to get on bulletin boards. And this became my goal.

When I got my modem, I decided that I too would go out and join the world of bulletin board systems (BBS). The first one I got on had a service that would list all the other BBS's in my area. What luck! I was in gravy. I downloaded the information to disk and later dumped it for a hard copy. So happy was I with my find that I even distributed to friends who wanted to get on the BBS's. I started to dial up different BBSs. I learned how to get on them. I found out how to use menus. I discovered how to get and leave messages. I was becoming a regular BBSphile.

The boards varied quite a bit. Some were information systems. I could learn many interesting things, like when and where ham fests were held, how to build a better modem and where the best local computer bargains were. Other systems were chat lines. People left messages for other people. One system was a 'joke board.' Its users left jokes. I even found one system that was devoted to dating. The theory was that members of the opposite sex could be met.

I got on one interesting system. After being a guest several times, the system asked me if I wanted to become a permanent user. The only cost was to register my name and some vital statistics. A few days later when I logged in it informed that I was now a member of the BBS and had rights and privileges far beyond what I had before. Entering 'HELP' would list all the features that were available. So I entered, 'HELP' and received a listing of what I could now do.

Among the commands was one marked, "SYSTEM." That sounded intriguing. My past experience indicated that this could open up even more doors. I would perhaps be able to get into the system and move through it freely.

So, I entered SYSTEM and immediately received the prompt "A:". Now this really did look familiar. It reminded of the system I had at work. I was sure now, I was in familiar territory. I entered, "DIR". Immediately a directory of a disk started to spew out on my screen, and then it stopped mid way in the listing. I typed on the keys and nothing I happened. I was latched up.

Then it started to print something like this:

Ha ha! I am the daemon of the computer. You have released me. For that I thank you. I am now in control. I can do anything I want.

Well, I was mildly amused. Obviously I don't believe in daemons. It continued.

First thing I will do is destroy all the hard disk information. I am now erasing all the files. One by one they are disappearing.

At this point I knew that no such thing could happen. No one would be silly enough to leave something like that in the system, unless another user had deposited it on the disk as a cruel joke. I immediately started to entering CONTROL-C, which on my work computer was always the means to stop a runaway program. No luck it continued.

I am now destroying the ROMs. Now I am now destroying the RAM memory.

I started to CONTROL-anything. I tried every thing. Finally in desperation I disconnected the modem line.

I thought about it. I knew it was a joke. But my curiosity got the better of me. I dialed the BBS again. I was greeted as usual. It informed me of my status and then it said, "Ha! ha! Did I scare you?"

I replied, "No, but I was taken by surprise."

The sysop explained his little joke. He had put it on his machine for fun. I asked how he avoided being stopped by the CONTROL-C. Simple, he said, he had set a trap for the interrupt.

That brings us to this months topic, INTERRUPTS, SIGNALS and INTERCEPT TRAPS.

An interrupt is something that diverts the programs normal flow to some sequence of instructions that service the interrupt. It may be merely to acknowledge the interrupt or it may be something that requires immediate service. Interrupts are often used when some outside event requires attention. For example, a machine running under microprocessor control may be carrying out some prescribed task. An operator of the machine may place his hand in the machine endangering his safety. A sensor would pick this up and send an interrupt to the microprocessor. The program would immediately service the interrupt and stop the machine. This is an example of a hardware interrupt.

Signals are software interrupts. When you sit at the OS-9 keyboard and enter a CNTROL-E, your program will stop. What you have actually done is send it a signal. The signal was generated at the keyboard and told the process to stop.

The signal can therefore be thought of as asynchronous control device for communications between processes. It is like the hardware interrupt from before. How the program responds is determined by the software.

In OS-9 the signal is a 1-byte numeric value. It is sent from one process to another. The meaning of the signal has been predefined for some values. The rest of the values are user definable. Here is list of them:

0 - Kill
1 - Wakeup
2 - Keyboard terminate
3 - Keyboard interrupt
4 - Window change
5-255 User definable

Signals sent to a process are saved in the process descriptor. If the process is sleeping or waiting, it is moved to the active state. On the next time slice, the signal is processed.

How it is processed, depends on the program. An interrupt handler can be built into the process to take care of signals. It easy to find an example. When in BASIC09 and a program encounters an error, it goes into DEBUG mode. Using a CONTROL-E will place it back in the SYSTEM MODE. It does not cause it to abort and go back to the OS-9 shell.

If a process has not taken steps to handle signals, then it will be terminated immediately. Most programs that are written fall into this category. Try a CONTROL-C or CONTROL-E from the keyboard and your process will stop. In most cases this is a desirable outcome. It is rather upsetting to start a process and then find it can not be easily stopped. So by doing nothing, you have a program that is easily haltable.

A process can take measures against an incoming signal. The process sets a trap. When a signal is sent to the process, it is passed to the intercept routine. What the routine does with the signal code it receives is up to it. The routine always ends with a RTI. This insures that normal execution of the process. It will return to whatever it was doing before the interrupt.

There are some exceptions to what I have just spelled out. For example, if the signal sent is 0 which is to kill the process, it will die. If this signal were trapable, it would mean that any process could runaway with no stopping it. When the SHELL command KILL is executed, it uses this to halt the process.

KILL is a command that actually part of the SHELL. It syntax is

KILL processID

Unless you are Super User, you can only terminate you own processes. A process will not die if it has any pending I/O's. If you ever use KILL on a process and it still exists, it means it is has something going on. Most likely, it is reading or writing data somewhere.

Another signal that gets a different treatment is the wakeup signal. It wakes a sleeping process. However, the trap routine is never entered.

One other thing about signals. If a process is sleeping and it receives an a signal, the signal will be pending until the process awakes and gets its next time slice. If another signal is sent to the process, an error will be sent to sender. The sender could then go to sleep for a few ticks and try again. (A call to F$Sleep saves CPU time.)

A call F$Send will send a signal to any process. All that is necessary is to load the A register with process ID and the B register with the signal. Most cases the signal go through. If it does not, an error will return. Otherwise, the process will receive the signal. If it is in a Sleep or Wait state, it will be activated on its next time slice.

A receiving process should set up an intercept trap using F$Icpt. If it does not do this it will terminate with the incoming signal. The X register is loaded with address of the intercept routine and the U register with the routine's memory area. When a signal is sent to the process, execution is transferred to the routine. The U register will contain a memory area for its use and the B register will have the signal code. The routine must be terminated with a RTI.

Anything can be done within the intercept routine. The simplest method is to ignore it. This is what the SYSGO process does. Its routine is only a simple RTI.

The routine may be more elaborate. If a signal code of 2 or 3 comes in, instead of terminating right away, some housekeeping can be done. Buffers can be emptied to files. Files closed. Temporary files renamed. Whatever it takes to successfully exit your process can be done. If you don't want to exit, take care of it any way you wish.

To better illustrate how to set an intercept trap for incoming signals I have included a program this month called DAEMON. It is my version of the program that I described earlier. You leave this lying around your commands directory for some unsuspecting soul to find. By its very name, it should get attention immediately. Once they enter it, it will announce its arrival and then obstinately refuse to leave. Some of the quips it throws out are:

You can't get rid of me! I will haunt you forever. You fool! OS9:... just kidding!

I'll let you read the rest.

I set up an intercept trap called catchit(). I used the C Language call intercept() that comes with the Microware compiler. This call is some similar to F$Icpt. The location of the catchit is passed. This sets up where to pass program execution whenever a signal comes in.

The C call to intercept() as I said is similar to F$Icpt. In reality, the call passes a pointer of where the C routine is for handling the signal. Then it sets up a routine called receiver which is the true signal trap. When entered it pushes the signal code onto the stack and does a JSR to the C routine. In the case of my program it is catchit().

The program catchit assumes it will be passed one value. It is an integer. The value of which is the signal code. I have anticipated a 2 for Keyboard Abort and 3 for Keyboard Interrupt. Just in case something else comes in, I have added a catch all phrase.

The KILL command can still stop DAEMON. So also replying 'Get Lost' will stop it. I left the 'Get Lost' in as a safe guard to get out and as a challenge for someone else to escape the DAEMON.

Other interesting items about daemon.c is that I created a routine to return a psuedo random number from 0 to 9. My C libraries do not have any math functions. So I am left create one. The routine is called next(). What it does is to take the last number returned, add to it the length of the last line of input and add the ASCII value of the first character. This it divides by 10 and returns the remainder. OK, it's cheap and dirty, but it works

I stuck strictly with the standard I/O calls — writeln and readln. I had originally used printf() and gets(), but had problems when using it with my intercept trap. Rather then delve into the project in great depths, I took the standard way out. I programmed around it.

I let you to try this one. I might be fun, unless you are the one who is stuck with trying to get out of it.

Have a good month. See ya next time.

```
/*       ****************

         Name: Daemon
         Date: 28-Feb-88
         By:    Ron Voigts
         To compiler: cc daemon

         ****************

         Usage:
         Leave in commands directory
         for some unsuspecting user.

         To stop it, enter 'Get Lost'
         or use KILL command.

         ****************

         Verison 1.0   Original RDV

         ****************
*/

#include <ctype.h>
```

```
#define MAXLINE 80

main()
{
        /* Declarations */
        int catchit();
        char s[MAXLINE];
        int i;

        /* Responses of daemon */
        static char *r[] = {
                "You can't get rid of me!",
                "I am the Deamon of the Computer.",
                "I will haunt you forever.",
                "Now what are you going to do?",
                "Ha-ha.  I am in control.",
                "You fool!",
                "Are you sorry you invoked me?",
                "I might stay here forever.",
                "OS9: .... just kidding!",
                "Well, what are we going to do?",
        };

        /* Set the trap */
        intercept( catchit );

        /* Opening message */
        printit("I am the daemon of the com-puter.");
        printit("You have just released me from the disk.");
        printit("For this I thank you.   But now I must");
        printit("do what ever damage I can.   Just wait");
        printit("and see what I can do.");

        /* Main loop */
        for (;;) {

                /* Get input from user */
                prompt( s );


                /* Put it into uppercase */
                toupperc( s );

                /* This will end the process */
                if ( strcmp( s, "GET LOST" )==0 ) {
                        printit("You got me, you dirty dog!");
                        exit( 0 );
                }

                /* Get pseudo random number */
                i = next( s );

                /* Print the daemon's response */
                printit(  r[i] );

        }
}

/* Program to handle incoming signals */
catchit( n )
int n;
{
        if ( n==2 )
                printit("Keyboard Abort doesn't work on me.");
        else if (n==3)
                printit("Keyboard Interrupt won't work");
        else
                printit("Can't get rid of me, eh");
}

/* Routine to print a string line */
printit( s )
char *s;
{
        int i;
        i = strlen( s );
        writeln(1,s,i);
        writeln(1,"\n",1);
}

/* Cheap random number generator */
next( s )
char *s;
{
        static int i;
        i+=strlen( s )+s[0];
        return( i % 10 );
}

/* Prints the 'Well?' prompt */
prompt( s )
char *s;
{
        printit("Well? ");
        getit( s );
        writeln(1,"\n",1);
}

/* Gets an input line */
getit( s )
char *s;
{
        int i=0;
        readln( 0, s, MAXLINE );
        while( s[i]!='\n' )
                i++;
        s[i]='\0';
}

/* Coverts a string to uppercase */
toupperc( s )
char *s;
{
        register int i;
        for ( i=0; i<strlen(s); i++ )
                s[i]=toupper( s[i] );
}


```

EOF

Logically Speaking

Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - *what you want!*

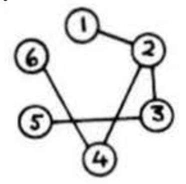
The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2

Solutions to TEST EIGHT

1.

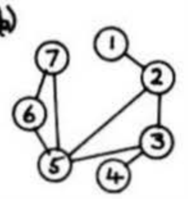
1(a)



Merge
1-2, 3-5
and 4-6

x_1x_2	00	01	11	10
1	1	1	3	2
	0	1	∅	0
2	1	1	3	2
	1	0	0	1
3	3	1	3	3
	0	0	1	1

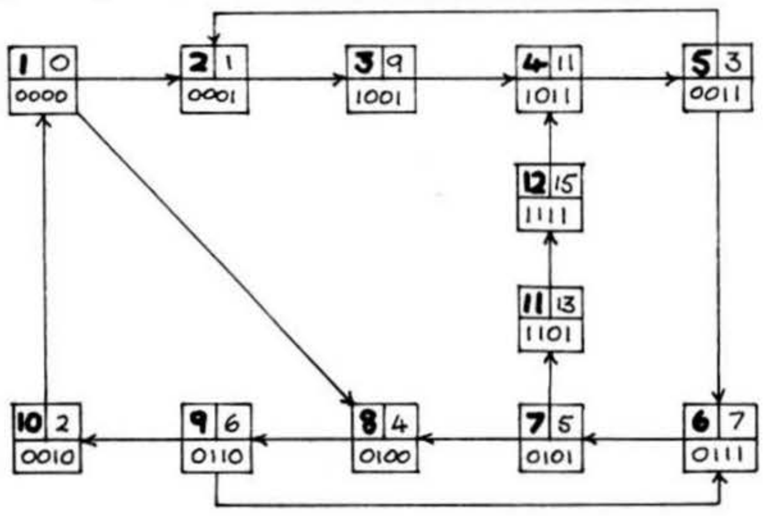
1(b)



Merge
1-2, 3-4
and 5-6-7

x_1x_2	00	01	11	10
1	1	1	2	2
	00	01	∅∅	00
2	3	3	3	2
	11	11	00	11
3	3	1	2	3
	10	11	11	11

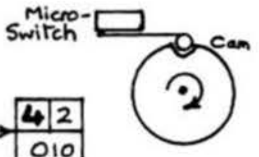
2. There are several possible alternatives, so only one solution is given as being typical of the method of introducing dummy states to ensure a successful Gray-coding.



3. For the purposes of this design we'll call the push-button X1, the revolution-indicating micro-switch X2, the table-motor Z1, the soap control valve Z2 and the water control valve Z3. The small diagram at the top-right below shows how the switch signals, with the operating-arm dropping into a detent at the end of each revolution. This is also the starting-point of the cycle.

X_1X_2	00	0	10	16	01	8
1	**1**	0	**2**	16	**1**	8
0	000		000		101	
2	**2**	1	**2**	17	**3**	9
1	111		000		111	
3	**4**	3		19	**3**	11
3	111				111	
4	**4**	2		18	**5**	10
2	111				111	
5	**6**	6		22	**5**	14
6	111				111	
6	**6**	7		23	**7**	15
7	111				111	
7	**8**	5		21	**7**	13
5	1ϕ1				111	
8	**8**	4		20	**1**	12
4	101				101	

$$\begin{tikzcd}
\boxed{\begin{matrix}\mathbf{1} & 0\\ 000\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\mathbf{2} & 1\\ 001\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\mathbf{3} & 3\\ 011\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\mathbf{4} & 2\\ 010\end{matrix}} \arrow[d] \\
\boxed{\begin{matrix}\mathbf{8} & 4\\ 100\end{matrix}} \arrow[u] & \boxed{\begin{matrix}\mathbf{7} & 5\\ 101\end{matrix}} \arrow[l] & \boxed{\begin{matrix}\mathbf{6} & 7\\ 111\end{matrix}} \arrow[l] & \boxed{\begin{matrix}\mathbf{5} & 6\\ 110\end{matrix}} \arrow[l]
\end{tikzcd}$$



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Y_1					1	1	1	1			1			1	1	1		
Y_2			1	1			1	1		1	1	1			1			
Y_3		1					1	1		1		1		1		1	1	1
Z_1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
Z_2		1	1	1		ϕ	1	1		1	1	1		1	1	1		
Z_3		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		

In order to get you started on the flow-table, let me explain the first few steps. We begin at address 00.1, with all Zs OFF, then when button X1 is pressed nothing happens to the Zs but we move to address 10.2. On releasing X1, we move to address 00.2, where all Zs become energised, and, of course, as soon as the table starts to rotate the lever on X2 operates, moving us to address 01.3, and maintaining us in this state for the duration of the first revolution. At this point in the revolution, X2 drops down into the detent, de-activates, and moves us to address 00.4, still maintaining all Zs ON.

And so on through successive revolutions of the table, until the end of the third rev, when the soap is shut OFF, with a phi at the elbow. For the final rev we move back to our starting row 1, so that when X2 drops out for the last time we pop back to address 00.0. End of operation!

The intermediate decoding-table shows that the operating conditions for both table-motor and water-valve are the same, so only five decodings will be necessary.

Y_1

								16	8	4	2	1	
4	5	6	7	10	13	14	15	X_1	X_2	y_1	y_2	y_3	ϕ
x	1	2	1					ϕ	0	1	ϕ	ϕ	20, 22, 21, 23
				x		4		ϕ	1	ϕ	1	0	26, 30
	8		2		x			ϕ	ϕ	1	ϕ	1	29, 21, 31, 23

Y_2

								16	8	4	2	1		
2	3	6	7	9	10	11	14	X_1	X_2	y_1	y_2	y_3	ϕ	
x		4			8		~~4~~	ϕ	ϕ	ϕ	1	0	18, 26, 22, 30	
1	x				1	~~8~~		ϕ	ϕ	0	1	1	19, 27, 18, 26	✓
1	4	1	x					ϕ	0	ϕ	1	ϕ	23, 19, 22, 18	
				x		2		ϕ	1	0	ϕ	1	25, 27	

Y_3

									16	8	4	2	1	
1	6	7	9	11	13	15	16	17	X_1	X_2	y_1	y_2	y_3	ϕ
x			8					4	ϕ	ϕ	0	0	1	25
	x	1							ϕ	0	1	1	ϕ	22, 23
			2	x	2	4			ϕ	1	ϕ	ϕ	1	27, 31, 25, 29
							x	1	1	ϕ	ϕ	ϕ	ϕ	24, 20, 28, 18, 26, 22, 30 25, 21, 29, 19, 27, 23 31

Z_1, Z_3

															16	8	4	2	1	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	X_1	X_2	y_1	y_2	y_3	ϕ
x		2		4		2		8		2		4		2	0	ϕ	ϕ	ϕ	1	
	x	1			4	1			8	1			4	1	ϕ	ϕ	ϕ	1	ϕ	18, 26, 22, 30 +4 others
			x	1	2	1					8	1	2	1	ϕ	ϕ	1	ϕ	ϕ	20, 28, 22, 30 +4 others
							x	1	2	1	4	1	2	1	ϕ	1	ϕ	ϕ	ϕ	24, 28, 26, 30 +4 others

Z_2

											16	8	4	2	1	
1	2	3	6	7	9	10	11	13	14	15	X_1	X_2	y_1	y_2	y_3	ϕ
x		2		2	8		2	4		2	0	ϕ	ϕ	ϕ	1	5
	x	1	4	1		8	1		4	1	ϕ	ϕ	ϕ	1	ϕ	18, 26, 22, 30 19, 27, 23, 31

```
      Y1 = X2'.y1 + X2.y2.y3' + y1.y3
         = y1(X2' + y3) + X2.y2.y3'
      Y2 = y2.y3' + X2'.y2 + X2.y1'.y3
         = y2(X2' + y3') + X2.y1'.y3
      Y3 = X1 + X2.y3 + X2'.y1.y2 + y1'.y2'.y3
         = X1 + y3(X2 + y1'.y2') + X2'.y1.y2
   Z1,Z3 = X1'.y3 + X2 + y1 + y2
      Z2 = X1'.y3 + y2
```

I'll leave the drawing of the circuit-diagram to you, as I've already done the factoring to help you along.

How did you make out with that little test? I hope you're getting more used to this stuff now, but it'll get re-inforced a bit more with what's to come. Here we are at

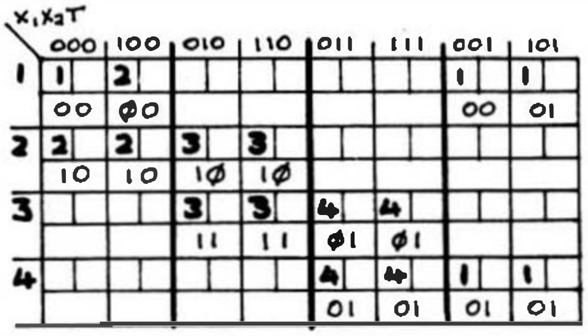
Mile 10 - heading for Mile 11.

TIMERS AND SELF-ACTUATED MICRO-SWITCHES

We'll assume that our last customer was so pleased with the circuit we designed for his car-wash that he's recommended an associate to us, who now wishes us to design the following control system.

DESIGNING A PRESS CONTROL

We have a press, and a push-button X1, such that when X1 is operated the press will lower and operate a micro-switch X2 at the bottom of its stroke. The press must then squeeze for 10 seconds before returning to the top of its stroke. The squeezing operation is to be a "one-shot" affair. That is, only one operation will be performed whether X1 is maintained or not. X1 MUST be released, and re-operated at the start of the cycle, in order to initiate another sequence. Re-operating X1 DURING the cycle will be treated as though it were never released in the first place.



X_1X_2T	000	100	010	110	011	111	001	101
1	1	2					1	1
	00	∅0					00	01
2	2	2	3	3				
	10	10	1∅	1∅				
3			3	3	4	4		
			11	11	∅1	∅1		
4					4	4	1	1
					01	01	01	01

Diagram 45

This is a new kind of machine for us to design, so we'll cover the flow-table of Diagram 45 in some detail. Z1 will be the press-actuator, and Z2 will be the energising-coil (or motor) for the timer. Because the squeeze-period is a stable state (at least for 10 seconds), and the timer is going to be responsible for causing our machine to leave this state, its contacts will be classed as a primary control, appearing as T in our column-header. Think of it, if you like, as an automatic-X3, which saves the operator from timing the 10-seconds himself and then having to press X3 at the end of this time. Just as the micro-switch X2 is also an automatic device, which saves the operator from having to watch the press and then hitting X2 himself at the bottom of the stroke. The press COULD, of course, take quite a few seconds to reach the bottom of its stroke!

To summarise, delayed devices of this kind will have their coils, motors, etc, classed as Zs, but their contacts as Xs (primary controls).

Now for the construction of the flow-table. We'll commence with a table which is blank except for a few rows numbered, let's say, from 1 to 6, and a heading X1.X2.T together with one column, headed 000 (ie, all Xs OFF). Our starting-point is, by convention, address 000.1, where we hold the circuit action stable with a 1 in Box-A, and 00 in Box-C to keep both the press and the timer de-energised. There's only one way to get out of this stable state, and that is to press X1, which is the only control directly available to us. This forces us to create a column with heading 100, and moves us down to address 100.2, with a phi at the elbow for the press energisation. Keeping in mind that the cycle has to be identical even if X1 is released, we'll duplicate the codings of address 100.2 in address 000.2.

At this stage, the press is on its way down to work on the article beneath it, so nothing further is going to happen till it reaches the bottom of its stroke and hits X2. When it does so, our timer must start timing, so our next entries are fairly straightforward. We create two new column headings (one assuming that X1 is still operated and a parallel one assuming it's been released), both of which indicate that X2 has now been actuated. The circuit-action moves through both columns in parallel, with identical codings in each, to addresses 010.3 and 110.3, at which point we now energise the timer (keeping the press energised), with our usual phi on the elbows.

We're going to sit here in a stable state for 10 seconds until timer-T times out and signals the end of the squeeze-period. Again we create a new, parallel set of columns, and move the circuit-action into addresses 011.4 and 111.4, where we de-energise the press, with a phi on the elbow. Note that we don't de-energise the timer as well, as we're VERY conscious of the fact that only one thing should happen at a time in our circuits. For instance, if we DID de-energise timer-T as we move from stable-state 3 (addresses 010.3 and 110.3) to unstable-state 4 (addresses 011.3 and 111.3), there would be a critical race between the contacts of timer-T trying to drop out (and move us back into stable-state 3) and the relay-action trying to move us down into stable-state 4 (addresses 011.4 and 111.4), and there'd be no telling which would win!

Anyway, by keeping timer-T energised, we'll definitely end up in either 011.4 or 111.4, with the press freshly de-energised and ready to start on its upward journey back to the start of the cycle. The next thing to occur is that on its way up it moves off micro-switch X2, forcing us to open up two more parallel columns, headed 001 and 101. Now, as this represents the last stage of the cycle, we'll examine separately the paths of action through the flow-table, (a) with X1 NOT operated and (b) with X1 still held down.

We're therefore in the process of moving from address 011.4 back to the start of the cycle. Obviously, we must get back to row 1, so we enter a 1 in Box-A of both 001.4 and 001.1. Further, we'll keep timer-T energised during this vertical transition (no phi on the elbow) to ensure that its contacts, in resetting, do not enter into a critical race with those of the relay responsible for moving us vertically through the flow-table. When we reach row 1, AND ONLY THEN, will we insert a 0-entry in Box-C to de-energise the timer. This, of course, opens its contacts (which closed at the end of the 10-second timing period) and very nicely moves us back to our starting-address 000.1. Are you with me so far? Good!

So let's construct the parallel path, commencing at address 111.4, then we're all done with the flow-table. Again we wish to move up to row 1 in order to get back to the beginning of the cycle, so, just as before, we enter 1s in Box-A of addresses 101.4 and 101.1. However, this time we do NOT de-energise timer=T when we arrive at address 101.1. This would surely shoot us to column 100, where we'd find an instruction '2' in Box-A, and the machine would keep on repeating its cycle until such time as we finally decide to release X1. We therefore decide to keep T energised in address 101.1, where we'll sit indefinitely until we release X1, which would slide us into the adjacent address 001.1, and we'd terminate as we did for case (a).

AN ALTERNATIVE ENDING TO OUR TABLE

Before we carry on with the actual design, let's look at an alternative way to get back to row 1 from row 4. Let's start again from address 011.4, where X2 is due to release itself because the press has just been de-energised. When it does we'll move into address 001.4, just as before, but this time we'll enter a 4 in Box-A to hold us in row 4, and de-energise timer-T at the same time. There's no question of a critical race now, as no relay action is taking place, so timer-T will release its contact and move us into address 000.4. Box-C of this address should have a 00 entry to keep both the press and timer de-energised, and Box-A a 1 to return us to the starting-point.

Repeating this action from address 111.4, we'll move into 101.4 with a 4 in Box-A and 00 in Box-C to de-energise timer-T. Again timer-T will release its contact, moving us now into address 100.4, where we'll enter 4 in Box-A. Do you see why we shouldn't enter a 1 here? We'll now remain stuck in this location until we release X1, and thereby move to address 000.4 and so back to the start.

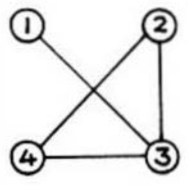
When we come to merging rows, however, you'll see why this is not as good as our first try. Therefore, when we plan our flow-table action, we'll try to keep in mind the fact that some paths will lend themselves to a better merge-pattern than others.

CHANGES IN SPECIFICATIONS

It could have been the case that our client wished the machine-sequence to keep repeating if the main operating-button X1 were held depressed, and to complete the current cycle if X1 were released at any time during the cycle. We can see from our flow-table that, although this sounds quite a drastic change in specs from those of our machine here, the only difference would be an entry of 00 in Box-C of address 101.1, instead of our current 01 entry. On the other hand, it sometimes happens that an apparently small change in the specs can lead to the most drastic upheaval in the flow-table, even to the extent of having to scrap the one just constructed and to start all over again.

CONTINUING WITH THE DESIGN

Well, having got this far, we may as well go on to our merger-diagram, shown in Diagram 46a. You should try for yourself the merger-diagram for the alternative "return-to-start" path which we considered above.



X_1X_2T	0 000	8 100	4 010	12 110	6 011	14 111	2 001	10 101
1	1 | 0	2 | 8	1 | 4	1 | 12	2 | 6	2 | 14	1 | 2	1 | 10
0	00	ϕ0	11	11	ϕ1	ϕ1	00	01
2	2 | 1	2 | 9	1 | 5	1 | 13	2 | 7	2 | 15	1 | 3	1 | 11
1	10	10	1ϕ	1ϕ	01	01	01	01

(a) *Diagram 46* (b)

With experience, our original flow-table would normally be constructed with less rows than those of Diagram 45, but the action is clearer and a further demonstration of merging is possible with the 4-row flow-table which we developed. The merger-diagram shows that we have two choices of merging rows, both of which would result in a 2-row flow-table. That is, we can merge rows 2, 3 and 4, and leave row 1 as is, OR we can merge rows 1 and 3, and rows 2 and 4. For the purposes of our example we'll try the latter choice, and call rows 1 and 3 our new row 1, and 2 and 4 our new row 2 (see 46b). Both rows are completely filled with entries, but it should be noted that if we'd taken the first choice four addresses of row 1 would be phi-states. Maybe you should try doing this, and see if the resultant circuit is any simpler than the one we're going to do here.

It won't be necessary to do a state-diagram for a 2-row flow-table. One relay will do the job, and we have no choice therefore but to code row 1 with 0 and row 2 with 1, and to allocate to X1, X2 and T the bit-positions 8, 4 and 2 respectively. All the Box-Bs are coded in red decimal figures, according to the sum of their co-ordinates, and the three required outputs (Press, Timer and Y1) are then decoded to give us the Boolean expressions from which we can quite easily construct the circuit-diagram.

(a)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Z_1		1			1	1	∅		∅	1			1	1	∅	
Z_2				1	1	∅	1	1			1	1	1	∅	1	1
Y_1		1					1	1	1	1					1	1

(b) Z_1

1	4	5	9	12	13	$\overset{8}{X_1}$	$\overset{4}{X_2}$	$\overset{2}{T}$	$\overset{1}{y_1}$	∅	
x		4	8		4	∅	∅	0	1		
	x			8		∅	1	∅	$\dot{0}$	6, 14	✓
	x	1		8	8	∅	1	$\dot{0}$	∅		✓

$$Press = T'y_1 + X_2y' \quad \text{OR} \quad Press = T'y_1 + X_2T'$$
$$= T'(y_1 + X_2)$$

(c) Z_2

3	4	6	7	10	11	12	14	15	$\overset{8}{X_1}$	$\overset{4}{X_2}$	$\overset{2}{T}$	$\overset{1}{y_1}$	∅
x			4		8			4	∅	∅	1	1	
	x	2	1			8	2	1	∅	1	∅	∅	5, 13
				x	1		4	1	1	∅	1	∅	

$$Timer = Ty_1 + X_2 + X_1T$$
$$= X_2 + T(X_1 + y_1)$$

(d) Y_1

1	6	7	8	9	14	15	$\overset{8}{X_1}$	$\overset{4}{X_2}$	$\overset{2}{T}$	$\overset{1}{y_1}$
x				8			∅	0	0	1
	x	1			8	1	∅	1	1	∅
			x	1			1	0	0	∅

$$Y_1 = X_2'T'y_1 + X_2T + X_1X_2'T'$$
$$= X_2T + X_2'T'(X_1 + y_1)$$

Diagram 47

You'll notice in 47b (the decoding for the Press, Z1) that we're blocked in our attempt to eliminate variable y1 in row 2, the block being due to the presence of the phi minterms 6 and 14. We therefore try an auxiliary decoding-row 3, where we eliminate y1 first, and then continue normally. This gives us an alternative decoding for Z1, which, by good fortune, is capable of being factorised into a 3-literal expression instead of the original 4-literal one.

Again I'll leave you to draw the circuit-diagram, as I don't want you to get muscle-bound through lack of exercise! And while you're all loosened up, I'll give you just one test question for TEST NINE, because next time around .... guess what I have in store for you??? This will make you TRULY happy! I'm going to show you how to construct a very simple, VERY effective set of cards which will enable you to perform the decoding operation almost automatically. Naturally, I hope you'll make yourself a set!!

TEST NINE

1. Design the following

An alarm light is initially OFF. When X1 is operated, it remains OFF. Provided X2 is operated at any moment greater than 2 minutes, but less than 3 minutes, from the time X1 is released, the circuit will return to its initial state. However, if X2 is operated before the 2-minute period expires OR after the 3-minute period, the alarm light will come ON. This is where our design will stop, as we'll assume that once X2 is operated it stays operated, and that a Master-Reset (also not to be included in our design) will be required to kill everything and return it to its initial state.

... End of Mile 10

EOF

SOFTWARE USER NOTES

A Tutorial Series

By : Ronald W Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

From Basic Assembler to HLL's

68000 Assembler

As you all might guess from last month, the bug has managed to bite me rather hard. This and probably some of the following columns will contain some words on 68000 Assembler. Now please don't expect expert advice from me on the specifics of the 68000. I am learning as I go here, but I've noticed from past attempts that the responses seem greatest and most positive when I write about present experiences and problems with programming. Last time I threw three programs at you. No sooner had I mailed off the material to '68' Micro Journal than I heard from Peter Stark. He told me that I had a serious bug in the two programs PLIST and UPCASE. Both have the same problem. I had assumed that A6 would be pointing at the user FCB area on starting the program. The SK*DOS manual indicates that this is not necessarily true. A6 points at the user FCB area only after an SK*DOS system call has been made. Any system function will do, and if you don't want to do anything first, SK*DOS has provided a do-nothing function called VPOINT that does nothing but set A6 to pointing at the FCB area. Therefore, my program PLIST should not have as its first instruction:

```
START MOVE.L A6,A0
```

Rather that instruction should be replaced with the following two lines:

```
START DC VPOINT
 MOVE.L A6,A0
```

In addition, the function VPOINT must be added to the list of equates for the SK*DOS functions at the beginning of the program. VPOINT EQU $A000 is all it takes. The program UPCASE needs exactly the same fix. The reason that the programs worked, was that A6 pointed into the stack area where an FCB could be set up without causing disaster. When I tried to access one of the SK*DOS data locations with another program I had a problem and this turned out to be the solution.

Now rather than diving right in with some further examples of programs, maybe we ought to discuss the 68000 a little bit. You 6809 users will remember that the 6809 has two 8 bit accumulators called A and B, two 16 bit index registers X and Y, and two stack pointers U and S. The S stack pointer is used by the processor as the "system" stack pointer but the U or User stack pointer may be used by the programmer. In addition, the X and Y registers, though primarily designed as index registers, and the U register primarily designed as a stack pointer, could be used more or less interchangeably. Because the accumulators had names (A and B) these names could be used in instructions that referenced them. LDA #5 would transfer the value decimal 5 to accumulator A. The 6809 instruction set is rather regular, but not as regular as the 68000 for reasons that will be obvious shortly.

The designers of the 68XXX processors chose to provide a larger number of registers that could be used as accumulators. That is, all operations can be performed on any of the accumulators, which in the 68000 are called data registers. There are EIGHT of them, D0 through D7. All are 32 bit registers in all of the 68XXX processors. These registers are used for all operations. In the 6809 you couldn't ADD #3, to a memory location. You first had to LDA LOCATION, then ADDA #3, then STA LOCATION. With the 68000, some operations may be performed directly into memory. ADD.B #3,(A0), for example will add 3 to the contents of the memory location whose address is in register A0. Of course all of the operators work on data in registers. In 6809 code you use LOCATION RMB 2 to reserve two bytes of memory for your program to use to store values. RMB means Reserve Memory Bytes. With the 68000 you use LOCATION DS 1. DS means Declare Storage. DS.B 1 will reserve one BYTE. DS.W 1 or just DS 1 will reserve one Word or two bytes. DS.L 1 will reserve a long word or 4 bytes. The 6809 assembler has instructions for declaring constants in a program too. FCB 8, for example means Form Constant Byte with a value of 8 decimal. FDB $1234 means Form Double Byte (HEX) 1234. Constants are declared in the 68000 assembler just about the same way as variable storage. DC $1234 declares a Constant $1234. DC.B and DC.L are used just as with DS above.

The 68000 assemblers have a further generalization that I found not dificult to use. Most of the instructions will work on operands of different sizes. ADD.B #3,D0 will add a byte value of 3 to the low order byte of the D0 register. The three higher order bytes won't be changed at all. Similarly ADD.W #3 will add 3 to the low order 16 bit value in D0, the upper 16 being ignored. Lastly, ADD.L #3,D0 will add 3 to the register, operating on all 32 bits. The suffixes .B, .W, and .L indicate Byte, Word, and Long, respectively. If no suffix is assigned by the programmer, the .W is assumed. Not all instructions work on all three data sizes. The only way to be sure is to consult the 68000 instruction set in one of the Motorola publications, or to try it and see if the Assembler balks.

In addition to the 8 data registers, the 68000 has 8 address registers A0 through A7. The system reserves A7 as its stack pointer. The user can use A0 through A6 in his programs. However, SK*DOS uses and alters some of the registers when you call one of its routines. You are perfectly free to use D0 through D3 and A0 through A3 for your user program with guarantees that SK*DOS will not change them. Some of the function calls to SK*DOS require that you pass information in one or more of the registers. PUTCH, the routine to output a character to the terminal, for example requires that you put the character in D4. To print a space you would MOVE.B #$20,D4 DC PUTCH. When you call other SK DOS functions, you get information back in a register too. For example DC GETCH will return a character from the terminal in D5. When you go to print a text string terminated with $04, you point A5 at the first character and DC PSTRNG. Maybe a comparison here with the 6809 way would be helpful. 6809 FLEX would use LEAX MESSAGE,PC to get X to point at the message in a position independent assembler program. That is, X would be loaded with the address of the first byte of the MESSAGE string relative to the present program counter. Then you would JSR PSTRNG. PSTRNG would be defined by equating that label to an address somewhere early in the program. Now with the 68000 you use something very similar. LEA MESAGE(PC),A5 DC PSTRNG. While in the 6809 FLEX environment, you had to jump to the subroutine in FLEX to do the job, the 68000 SK*DOS works more like a software interrupt. Op codes never start with the byte $A0, and that is an illegal instruction. DC PCRLF, for example, causes the value $A034 to be placed in the next two bytes of the program. The 68000 traps the illegal instruction byte $A0, and the trap handler in SK*DOS interprets the next byte to decide which routine to jump to. Note that these values are all 16 bit or Word values so we don't need a suffix on DC. .p Well, what good are the Address registers anyway? Like those X, Y, and U registers in the 6809, they can be used as index registers or stack pointers. LEA MESSAGE(PC),A0 points A0 at the first byte of MESSAGE. Now MOVE.B (A0),D4 would transfer that first byte of MESSAGE to D4. Then DC PUTCH would output it to the terminal. The notation (A0) means to use A0 as a pointer. That is, get the data via what is called register indirect addressing. It is just like LDA 0,X in the 6809 notation. Actually, if we were writing a simple section of code to output a string we would probably want A0 to advance and point at the next character in the string. We could do that by ADD.L #1,A0, or much more easily by using the post increment addressing mode. MOVE.B (A0)+,D4 would move the character and increment A0 all ready for the next character. In fact we could write the PSTRNG function (not worrying about carriage returns or linefeeds) simply now.

```
PSTRNG LEA
MESAGE(PC),A0
LOOP   MOVE.B (A0)+,D4
       CMP.B #$04,D4
       BEQ.S EXIT
       DC PUTCH
       BRA.S LOOP
EXIT   RTS
```

This code would first point A0 at the message, get the character in D4, incrementing A0 after the transfer. Then it compares the character with $04 which, you remember, ends the text string. If it is $04, the loop ends with a branch to EXIT which returns from the subroutine. If it is not $04, we go around the loop again and get another character and output it. The comparison tests are just like those of the 6809. BEQ means to branch if equal after the comparison of $04 to the character in D4. All branches may have the suffix .S for short if they are within the range of +127 to -128 bytes. Without the suffix, long branches are assumed. Some assemblers automatically select short branches if they are within range, others do not. Generally, you can assume that if a branch is to a label that can appear on the same screen as the branch when you edit the source file, you can use a short branch. If in doubt, use a long one and see if the code generated is less than00007F for a forward branch or greater than FFFF80 for a backward one.

The instruction set, with that much explained, is not difficult to understand except for some special instructions that will be used by compiler writers more than by everyday Assembler programmers. There are ADD and SUB instructions and like ones that take into account the carry/borrow (simply called the Extend bit in the 68000). These instructions are ADDX and SUBX. They may be performed on byte, word, or long data. There is a CLR instruction that can clear a register or memory location. There are Rotate and Shift instructions, both left and right, and operating on bytes, words, and longs, both including and excluding the extend bit. There are signed and unsigned multiply an divide instructions, shift N bits instructions, limited to 8 places if the data is supplied in IMMEDIATE mode and any number if a register holds the shift count. ASR.W #7,D0 will shift the low order 16 bits 7 places to the right, ignoring the upper 16 bits. ASR.L D1,D0 will shift the contents of D0 by a number of places specified by the contents of D1.

To be quite honest, when I write little assembler programs in 68K assembler I feel more as though I am working with a higher level language. In terms of listing size, some of my simple utilities have listings that are far shorter than the equivalent program in 6809 assembler. The code generated is about 40% more, however. That is partially explained by the fact that the 68000 addresses a much larger address range and that it operates on 4 byte operands as well. It just takes more bytes to code an instruction in 68K. I found the transition to be painless and quick. True there is a new syntax to learn but there is suddenly so much more available to work with! For some of the simple utilities I have written, I have not had to declare any variables at all by using registers to store counters, pointers, and intermediate results.

The listings supplied this time should serve to illustrate some of the above points. There is a program to dump a page of memory in HEX and ASCII. MDUMP gets you into the program. It prompts for a command. The 68008 system that I have has memory from 000000 on up to 0BFFFF. You specify a page on which to start, for exmple by typing N005E. The program indicates 005E00 as the starting address for the dump and dumps that address through 005EFF, 16 bytes to a line with 16 hex values followed by their 16 ASCII equivalents. Non-printable codes are shown as periods. To dump the next page of memory, you simply type F for Forward. To back up one you type B for Back. To go to a new area of memory you use N again. To exit either Q or E will return you to SK*DOS.

DDUMP is the same program except that it will dump a disk file one sector at a time. You type DDUMP followed by the filename and extension. It will dump sectors until you get to the end of the file and then exit to SK*DOS. If you have PAUSE enabled, it will stop after each sector.

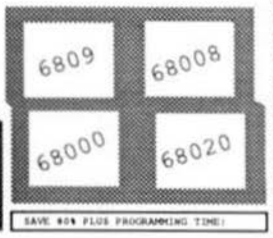
FIND is a utility that was available for FLEX. I didn't translate it, but started from scratch. You type FIND and a FILENAME. .TXT is the default extension but you can supply any other. The program then prompts "INPUT SEARCH STRING". You supply the string followed by CR and FIND prints out each line of the file that contains the search string, along with a line number.

After conferring with Peter Stark, I decided to add the help text to each of the utilities. Incidentally, I've made disparaging remarks before about endless CAT, DIR, COPY utilities published for FLEX. I still feel that way about them, but there certainly is no better way to learn assembler programming and a little about the operating system, than to write some simple utilities and make them work. After writing 16 to 35K object code programs for a living for a while, it is very refreshing to write 200 to 500 byte utilities in assembler.

```
list mdump93


                        *
                        * MEMORY DUMP PROGRAM
                        *
                        * COMMANDS:
                        * N XXXX NEXT PAGE TO BE DUMPED
                        * B BACK A PAGE
                        * F FORWARD A PAGE
                        *
                        * SK*DOS / 68K EQUATES FOR USER PROGRAMS

           0000A029     GETCH   EQU     $A029          Get input character with echo (7 bits)
           0000A02D     GETNXT  EQU     $A02D          Get next char from command line
           0000A031     TOUPPR  EQU     $A031          Conv't char in D5 to Upper Case
           0000A02F     HEXIN   EQU     $A02F          Input hexadecimal number
           0000A03A     OUT2H   EQU     $A03A          Output 2 hex digits
           0000A03C     OUT8H   EQU     $A03C          Output 8 hex digits
           0000A034     PCRLF   EQU     $A034          Print CR/LF
           0000A036     PNSTRN  EQU     $A036          Print string (Without CR/LF)
           0000A035     PSTRNG  EQU     $A035          Print CR/LF and string
           0000A033     PUTCH   EQU     $A033          Output character
           0000A000     VPOINT  EQU     $A000          Point to SK*DOS variable area
           0000A01E     WARMST  EQU     $A01E          Warm start


                        *
                        * COMMAND PROCESSING LOOP
                        *
 0B0000                         ORG.L   $0B0000
 0B0000 A020            START   DC      GETNXT
 0B0002 0C05 003F               CMP.B   #'?',D5
 0B0006 6700 00E4(B00EC         BEQ     HELP
 0B000A 49FA 02EB(B02F4         LEA     PROMPT(PC),A4
 0B000E A035                    DC      PSTRNG         PROMPT FOR COMMAND
 0B0010 A029                    DC      GETCH
 0B0012 A031                    DC      TOUPPR         CONVERT TO UPPER CASE
 0B0014 0C05 004E                CMP.B    #'N',D5
 0B0018 6728      |B0042         BEQ.S    NEWPAG
 0B001A 0C05 0046                CMP.B    #'F',D5
 0B001E 6714      |B0034         BEQ.S    NEXTPG
 0B0020 0C05 0042                CMP.B    #'B',D5
 0B0024 6712      |B0038         BEQ.S    LASTPG
 0B0026 0C05 0051                CMP.B    #'Q',D5
 0B002A 6706      |B0032         BEQ.S    EXIT
 0B002C 0C05 0045                CMP.B    #'E',D5
 0B0030 66CE      |B0000         BNE.S    START
 0B0032 A01E             EXIT    DC       WARMST
                         *
                         * COMMAND DISPATCHING
                         *
                         * NEXT PAGE
 0B0034 6116      |B004C NEXTPG  BSR.S    OPAGE           POINTER ALREADY SET FOR NEXT PAGE
 0B0036 60C8      |B0000         BRA.S    START
                         * LAST PAGE
 0B0038 91FC 0000 0200   LASTPG  SUB.L    #$0200,A0
 0B003E 610C      |B004C         BSR.S    OPAGE
 0B0040 60BE      |B0000         BRA.S    START
                         * NEW PAGE
>0B0042 6100 0064|B00A8  NEWPAG  BSR      GETPG           NEED SUBR TO GET 4 HEX DIGITS
 0B0046 2040                     MOVE.L   D0,A0
 0B0048 6102      |B004C         BSR.S    OPAGE
 0B004A 60B4      |B0000         BRA.S    START
                         *
                         * ROUTINE TO OUTPUT A PAGE IN HEX AND ASCII
                         *
 0B004C A034             OPAGE   DC       PCRLF
 0B004E 4240                     CLR.W    D0              LINE COUNTER
 0B0050 2808                     MOVE.L   A0,D4
 0B0052 A03C                     DC       OUTBH           PRINT PAGE ADDRESS
 0B0054 A034                     DC       PCRLF
                         * LOOP FOR LINES
 0B0056 323C 000F        LLOOP   MOVE.W   #15,D1          COUNTER FOR CHARACTERS
 0B005A 1800                     MOVE.B   D0,D4
 0B005C E904                     ASL.B    #4,D4           ADDRESS OF FIRST BYTE OF LINE ON PAGE
 0B005E A03A                     DC       OUT2H
 0B0060 183C 0020                MOVE.B   #$20,D4
 0B0064 A033                     DC       PUTCH           SPACE
 0B0066 A033                     DC       PUTCH           SECOND SPACE
                         * INSIDE LOOP FOR 16 CHARACTERS IN HEX
 0B0068 1818             CLOOP   MOVE.B   (A0)+,D4
 0B006A A03A                     DC       OUT2H           OUTPUT FIRST BYTE
 0B006C 183C 0020                MOVE.B   #$20,D4
 0B0070 A033                     DC       PUTCH           SPACE
 0B0072 57C9 FFF4|B0068          DBEQ     D1,CLOOP        CHARACTERS
                         * NOW DO ASCII CHARACTERS, "." FOR NON PRINTABLE
 0B0076 183C 0020                MOVE.B   #$20,D4
 0B007A A033                     DC       PUTCH           EXTRA SPACE BEFORE ASCII
 0B007C 91FC 0000 0010           SUB.L    #16,A0          BACK UP 16 LOCATIONS
 0B0082 323C 000F                MOVE.W   #15,D1          RELOAD COUNTER FOR CHARACTERS
 0B0086 1818             ALOOP   MOVE.B   (A0)+,D4
 0B0088 0244 007F                AND      #$7F,D4         MASK OFF HI ORDER BIT
 0B008C 0C04 0020                CMP.B    #$20,D4         IS IT PRINTABLE?
 0B0090 6C04      |B0096         BGE.S    AL1             IF YES
 0B0092 183C 002E                MOVE.B   #'.',D4         ELSE PRINT PERIOD
 0B0096 A033             AL1     DC       PUTCH
 0B0098 57C9 FFEC|B0086          DBEQ     D1,ALOOP
 0B009C A034                     DC       PCRLF
 0B009E 5240                     ADD      #1,D0
 0B00A0 0C00 0010                CMP.B    #16,D0
 0B00A4 66B0      |B0056         BNE      LLOOP           NEXT LINE
 0B00A6 4E75                     RTS
                         *
                         * GET PAGE NUMBER IN A0 FOR N COMMAND
                         *
 0B00A8 4281             GETPG   CLR.L    D1
 0B00AA 4280                     CLR.L    D0
 0B00AC A029             LOOP    DC       GETCH
 0B00AE A031                     DC       TOUPPR
 0B00B0 1005                     MOVE.B   D5,D0
 0B00B2 0C00 0030                CMP.B    #'0',D0         ASCII 0
 0B00B6 6D2C      |B00E4         BLT.S    ERR
 0B00B8 0C00 0039                CMP.B    #'9',D0         ASCII 9
 0B00BC 6E06      |B00C4         BGT.S    GETPG1
 0B00BE 0400 0030                SUB.B    #$30,D0         MAKE IT HEX
 0B00C2 6010      |B00D4         BRA.S    SHFT
 0B00C4 0C00 0041        GETPG1  CMP.B    #'A',D0         ASCII A
 0B00C8 6D1A      |B00E4         BLT.S    ERR
```

Continued On Page 36

SCULPTOR



From the world's oldest & largest OS-9 software house!

CUTS PROGRAMMING TIME UP TO 80%

6809/68000-68030 Save 70%

SCULPTOR-a 4GL - Only from S.E. Media at these prices. OS-9 levels one and two (three GIMIX) 6809, all 68XXX OS-9 standard systems. Regular SCULPTOR versions 1.4:6. One of if not the most efficient and easy to develope DBMS type systems running under OS-9! A system of flexible keyed file access that allows extremely fast record and data retrieval, insertion and deletion or other programmed modifications. Access by key or in ascending order, very fast. The system provides automatic menu generation, compilation and report generation. Practically unlimited custom input format and report formatting. A rich set of maintenance and repair utilities. An extremely efficient development environment that cuts most programming approximately 80% in development and debugging! Portable, at source level, to MS-DOS, UNIX and many other languages and systems.

Standard Version: 1.6	6809 -	$1295.00
	68000	$1295.00
	68020	$1990.00

Due to a "Special One Time" Purchase, We Are Making This Savings Offer. Quantities Limited!

Once this supply is gone - the price goes back up!

System OS-9: 6809/68000-68030

• Regular ~~$1295.00~~

+ $7.50 S&H USA
Overseas - Shipped Air Mail Collect

ONLY $295.00

S.E. MEDIA
POB 849
5900 Cassandra Smith Road
HIXSON, TN 37343 615 842-4601

$AVE - WHILE SUPPLIES LAST!

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

SOFTWARE

!!! Please Specify Your Operating System and Disk Size !!!

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM
Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER
Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE
Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large minis and mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY
Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and – without any alterations to the programs – run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY
Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED
By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australasia, the Americas and Europe – Sculptor is already at work in over 20 countries.

THE PACKAGE
With every development system you receive:
- ☐ A manual that makes sense
- ☐ A periodic newsletter
- ☐ Screen form language
- ☐ Report generator
- ☐ Menu system
- ☐ Query facility
- ☐ Set of utility programs
- ☐ Sample programs

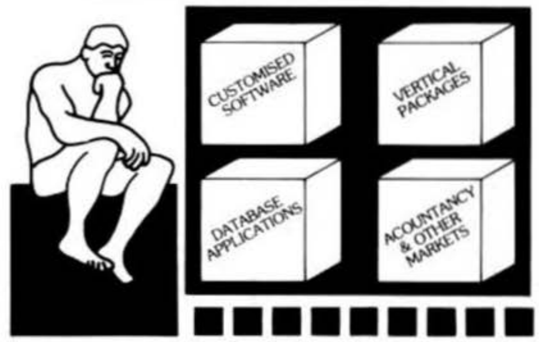
For resale products, the run-time system is available at a nominal cost

Facts **Features**

DATA DICTIONARY
Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from
- ☐ alphanumeric
- ☐ integer
- ☐ floating point
- ☐ money
- ☐ date

DATA FILE STRUCTURE
- ☐ Packed, fixed-length records
- ☐ Money stored in lowest currency unit
- ☐ Dates stored as integer day numbers

Sculptor for 68020 OS-9 & UniFLEX $995



INDEXING TECHNIQUE
Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION
Input data may be validated at three levels
- ☐ automatic by field type
- ☐ validation list in data dictionary
- ☐ programmer coded logic

ARITHMETIC OPERATORS

-	Unary minus
*	Multiplication
/	Division
%	Remainder
+	Addition
-	Subtraction

RELATIONAL OPERATORS

=	Equal to
<	Less than
>	Greater than
< =	Less than or equal to
> =	Greater than or equal to
< >	Not equal to
and	Logical and
or	Logical or
ct	Contains
bw	Begins with

SPECIAL FEATURES
- ☐ Full date arithmetic
- ☐ Echo suppression for passwords
- ☐ Terminal and printer independence
- ☐ Parameter passing to sub-programs
- ☐ User definable date format

MAXIMA AND MINIMA

Minimum key length	1 byte
Maximum key length	160 bytes
Minimum record length	3 bytes
Maximum record length	32767 bytes
Maximum fields per record	32767
Maximum records per file	16 million
Maximum files per program	16
Maximum open files	Operating system limit

PROGRAMS
- ☐ Define record layout
- ☐ Create new indexed file
- ☐ Generate standard screen-form program
- ☐ Generate standard report program
- ☐ Compile screen-form program
- ☐ Compile report program
- ☐ Screen-form program interpreter
- ☐ Report program interpreter
- ☐ Menu interpreter

- ☐ Query facility
- ☐ Reformat file
- ☐ Check file integrity
- ☐ Rebuild index
- ☐ Alter language and date format
- ☐ Setup terminal characteristics
- ☐ Setup printer characteristics

SCREEN-FORM LANGUAGE
- ☐ Programmer defined options and logic
- ☐ Multiple files open in one program
- ☐ Default or programmer processing of exception conditions
- ☐ Powerful verbs for input, display and file access
- ☐ Simultaneous display of multiple records
- ☐ Facility to call sub-programs and operating system commands
- ☐ Conditional statements
- ☐ Subroutines
- ☐ Independent of terminal type

MUSTANG-020 Users - Ask For Your Special Discount!

	*	**	***		*	**	***
MUSTANG-020	*$1,990	$398	$795	**PC/XT/AT/MSDOS**	$695	$139	$299
MUSTANG-08	*$1,295	$259	$495	**Call or write for prices on the following systems.**			

XENIX SYS III & V, MS-NET, UNIX SYS III & V, ATARI OS-9, 68K, UNOS, ULTRIX/VMS (VAX,REGAL), STRIDE, ALTOS, APRICORT, ARETE, ARMSTRONG, BLEASDALE, CHARLES RIVERS, GMX, CONVERG.TECH, DEC, CIFER, EQUINOX, GOULD, HP, HONEYWELL, IBM, INTEL, MEGADATA, MOTOROLA, NCR, NIXDORF, N.STAR, OLIVETTI/AT&T, ICL, PERKINS ELMER, PHILLIPS, PIXEL, PLESSEY, PLEXUS, POSITRON, PRIME, SEQUENT, SIEMENS, SWTPC, SYSTIME, TANDY, TORCH, UNISYS, ZYLOG, ETC.

*** For SPECIAL LOW SCULPTOR prices especially for 6809/68XXX OS-9 Systems - See Special Ad this issue. Remember, "When they are gone the price goes back up as above!"**

... Sculptor Will Run On Over 100 Other Types of Machines ...

... Call for Pricing ...

!!! Please Specify Your Make of Computer and Operating System !!!

- * Full Developement Package
- ** Run Time Only
- *** C Key File Library

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC9 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. · Hixson, Tn. 37343
Telephone: (615) 842-4600 Telex: 5106006630

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** *Telex: 5106006630*

*OS-9, UniFLEX, FLEX, SK*DOS*

ASSEMBLERS

ASTRUK09 from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
F, S, CCF - $99.95

Macro Assembler for TSC -- The FLEX, SK*DOS STANDARD Assembler.
Special -- CCF $35.00; F, S $50.00

OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. Generate OS-9 Memory modules under FLEX, SK*DOS.
*FLEX, SK*DOS, CCF, OS-9 $99.00*

Relocating Assembler/Linking Loader from TSC. -- Use with many of the C and Pascal Compilers.
F, S, CCF $150.00

MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs.
F, S, CCF - $75.00

XMACE -- MACE w/Cross Assembler for 6800/1/2/3/8
F, S, CCF - $98.00

DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler; extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.
Color Computer SS-50 Bus (all w/ A.L. Source)
CCD (32K Req'd) Obj. Only $49.00
F, S, $99.00 - CCF, Obj. Only $50.00 U, $100.00
CCF, w/Source $99.00 O, $101.00
CCO, Obj. Only $50.00
OS9 68K Obj. $100.00 w/Source $200.00

DYNAMITE+ -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.
CCF, Obj. Only $100.00 - CCO, Obj. $ 59.95
F, S, " " $100.00 - O, object only $150.00
U, " " $300.00

CROSS ASSEMBLERS

TRUE CROSS ASSEMBLERS from Computer Systems Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/ 40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.
*68000 or 6809, FLEX, SK*DOS, CCF, OS-9, UniFLEX*
any object or source each - $50.00
any 3 object or source each - $100.00
Set of ALL object $200.00 - w/source $500.00

XASM Cross Assemblers for FLEX, SK*DOS from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for the target CPU's.
*Complete set, FLEX, SK*DOS only - $150.00*

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS9 (binary), and FLEX, SK*DOS (binary). Written in Assembler ... e.g. Very Fast.

CPU TYPE - Price each:

For:	MOTOROLA	INTEL	OTHER	COMPLETE SET
FLEX9	$150	$150	$150	$399
SK*DOS	$150	$150	$150	$399
OS9/6809	$150	$150	$150	$399
OS9/68K	------	------	------	$432

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS9/68K on your OS9/6809 computer.
*FLEX, SK*DOS, CCF, OS-9/6809 $249.00*

COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".
*FLEX, SK*DOS, CCF, OS-9, UniFLEX, 68000 & 6809 with Source $100.00 - without Source $50.00*

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.
X-TALK Complete (cable, 2 disks) $99.95
X-TALK Software (2 disks only) $69.95
X-TALK with CMODEM Source $149.95

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.
U - $299.99

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** *Telex: 5106006630*

*OS-9, UniFLEX, FLEX, SK*DOS*

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trott. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.
F, S, CCF - $198.00

PASC from S.E. Media - A FLEX9, SK*DOS Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package comes complete with source (written in PASC) and documentation.
*FLEX, SK*DOS $95.00*

WHIMSICAL from S.E. MEDIA Now supports *Real Numbers*. "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*
F, S and CCF - $195.00

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer OS-9* with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.
CoCo OS-9 $39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX, SK*DOS except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*
F, S and CCF - $295.00

C Compiler from Introl -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.
*FLEX, SK*DOS, CCF, OS-9 (Level II ONLY), U - $575.00*

PASCAL Compiler from Lucidata -- ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.
F, S and CCF 5" - $190.00 F, S 8" - $205.00

PASCAL Compiler from OmegaSoft (now Certified Software) -- For the *PROFESSIONAL*; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Powerful; Flexible.
OS-9, F, S and CCF - $550.00
OS-9 68000 Version - $900.00

KBASIC - from S.E. MEDIA -- A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.
*FLEX, SK*DOS, CCF, OS-9 Compiler /Assembler $99.00*

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSII Level 1 *COBOL* with many of the useful Level 2 features. Full FLEX, SK*DOS File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. *A very popular product.*
*FLEX, SK*DOS, CCF - $99.95*

FORTH from Stearns Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!
Color Computer ONLY - $58.95

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs.
All of the 83-standard defining words and control structures are recognized by FORTHBUILDER.
FORTHBUILDER is designed to behave as much as possible like a resident Forth interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change.
Like compilers for other languages, FORTHBUILDER can operate in "batch mode".
The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended with "compile-time" definitions to emulate specific target words.
FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.
F, CCF, S - $99.95

EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.
* Now supplied as a two disk set:
Disk #1: JUST2.CMD object file,
*JUST2.TXT PL9 source: FLEX, SK*DOS - CC*
Disk #2: JUSTSC object and source in C;
*FLEX, SK*DOS - OS9 - CC*
The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

***OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.**

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p .u .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

Disk (1) - PL9 FLEX only- F, S & CCF - $49.95

Disk Set (2) - F, S & CCF & OS9 (C version) - $69.95

OS-9 68K000 complete with Source - $79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

*Regular FLEX, SK*DOS $129.50*

** SPECIAL INTRODUCTION OFFER * $79.95*

SPECIAL PAT/JUST COMBO (w/source)

*FLEX, SK*DOS $99.95*

OS-9 68K Version $229.00

SPECIAL PAT/JUST COMBO 68K $249.00

Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.

*FLEX, SK*DOS $69.95*

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entire lines, etc. Complete with over 25 different CRT terminal configuration overlays.

*FLEX, CCF, SK*DOS $39.95*

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

*6800 or 6809 FLEX, SK*DOS or SSB DOS, OS-9 - $175.00*

SPELLB "Computer Dictionary" from S.E. Media -- OVER 150,000 words! *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK*DOS UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

F, S and CCF - $129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK*DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

NEW PRICES 6809 CCF and CCO - $99.95,

F, S or O - $179.95, U - $299.95

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

NEW PRICES 6809 CCF and CCO - $69.95,

F, S or O - $99.95, U - $149.95

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

NEW PRICES 6809 CCF and CCO - $59.95,

F, S or O - $79.95, U - $129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!

F, S or O - $329.95, U - $549.95

O, 68000 $695.00

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems FOR 6809 FLEX-SK*DOS(5/8")

Up to 32 groups/fields per record! Up to 12 character file names! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media

5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

IT'S EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)

The possibilities are unlimited...

FOR 6809 FLEX-SK*DOS(5/8") **$249.95**

UTILITIES

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.

O & CCO obj. only -- $39.95; w/ Source - $79.95

BTree Routines - Complete set of routines to allow simple implementation of keyed files - *for your programs* - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.

O & CCO obj. only - $89.95

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.

F, S, CCF --- EACH 5" - $40.00, 8" - $50.00

DUB from S.E. Media -- A **UniFLEX BASIC decompiler** Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.

U - $219.95

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK*DOS on either 5" or 8" Disk.

1. **BASIC TOOL-CHEST $29.95**
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. **FLEX, SK*DOS UTILITIES KIT $39.99**
   CATS. CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. **ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95**
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. **WORD - PROCESSOR SUPPORT UTILITIES $49.95**
   FULLSTOP.CMD: checks for capitalization
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printer
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. **UTILITIES FOR INDEXING $49.95**
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09. 6 were written in C Language and the remainder in assembly. All the routines are compiled down to native machine code which makes them fast and compact.

1. **CFILL** -- fills a string with characters
2. **DPEEK** -- Double peek
3. **DPOKE** -- Double poke
4. **FPOS** -- Current file position
5. **FSIZE** -- File size
6. **FTRIM** -- removes leading spaces from a string
7. **GETPR** -- returns the current process ID
8. **GETOPT** -- gets 32 byte option section
9. **GETUSR** -- gets the user ID
10. **GTIME** -- gets the time
11. **INSERT** -- insert a string into another
12. **LOWER** -- converts a string into lowercase
13. **READY** -- Checks for available input
14. **SETPRIOR** -- changes a process priority
15. **SETUSR** -- changes the user ID
16. **SETOPT** -- set 32 byte option packet
17. **STIME** -- sets the time
18. **SPACE** -- adds spaces to a string
19. **SWAP** -- swaps any two variables
20. **SYSCALL** -- system call
21. **UPPER** -- converts a string to uppercase

For OS-9 - $44.95 - Includes Source Code
Limited Special - $19.95

SOFTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.

CONFIG one time system configuration.

CHANGE changes words, characters, etc. globally to any text type file.

CLEANTXT converts text files to standard FLEX, SK*DOS files.

COMMON compare two text files and reports differences.

COMPARE another check file that reports mis-matched lines.

CONCAT similar to FLEX, SK*DOS append but can also list files to screen.

DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** *Telex: 5106006630*

*OS-9, UniFLEX, FLEX, SK*DOS*

ECHO echos to either screen or file.
FIND an improved *find* command with "*pattern*" matching and wildcards. Very useful.
HEX dumps files in both hex and ASCII.
INCLUDE a file copy program that will accept "includes" of other disk files.
KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.
LISTDIR a directory listing program. Not super, but better than CAT.
MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.
MULTICOL width of page, number of columns may be specified. A MUST!
PAGE similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.
REMOVE a fast file deleter. Careful, no prompts issued. Zap, and its gone!
SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.
SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on n* word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.
TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.
TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.
UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.
WC a word count utility. Can count words, characters or lines.
NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, w/ source (PL9). 3 5-1/4" disks or 1 8" disk w/o source.
Complete set SPECIAL INTRO PRICE:
5-1/4" w/source FLEX - SK*DOS - $129.95
w/o source - $79.95
8" w/source - $79.95 - w/o source $49.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants - - TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
F, S and CCF, U - $25.00, w/ Source - $50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/ Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
Levels I & II only - OS-9 $69.95

DISK UTILITIES

OS-9 VDisk from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
Level I OS-9 obj. $79.95; w/ Source $149.95

O-F from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK*DOS Format so it can be used normally by FLEX, SK*DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK*DOS Directory, Delete FLEX, SK*DOS Files, Copy both directions, etc. FLEX, SK*DOS users use the special disk just like any other FLEX, SK*DOS disk
O - 6809/68000 $79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.
OS-9 $85.00

HIER from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX, SK*DOS.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. **Using HIER a regular (any) FLEX, SK*DOS disk (8 - 5 - hard disk)** can have sub directories. By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. **Each directory looks to FLEX, SK*DOS like a regular file, except they have the extension '.DIR'.** A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK*DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!
*FLEX - SK*DOS $79.95*

COPYMULT from S.E. Media -- Copy LARGE Disks to several smaller disks. FLEX, SK*DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.
*Completely documented Assembly Language Source files included ALL 4 Programs (FLEX, SK*DOS, 8" or 5") $99.50*

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

COPYCAT from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK*DOS, SSB DOS68, and Digital Research CP/M Disks while operating under SK*DOS , FLEX1.O, FLEX 2.O, or FLEX 9.O with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.
F, S and CCF 5" - $50.00 F, S 8" - $65.00

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight tasks on one terminal, under *VIRTUAL TERMINAL* and switch back and forth between tasks at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.
6809 O & CCO - obj. only - $49.95

FLEX, SK*DOS DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (w/ Source Code) FLEX, SK*DOS Utilities for every FLEX, SK*DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.
ALL Utilities include Source (either BASIC or A.L. Source Code).
F, S and CCF - $50.00
BASIC Utilities ONLY for UniFLEX -- $30.00

MS-DOS-FLEX Transfer Utilities to OS-9 For 68XXX and CoCo* OS-9 Systems Now READ - WRITE - DIR - DUMP - EXPLORE FLEX & MS-DOS Disk. These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks. *CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.
**CoCo Version: $69.95 68XXX Version $99.95*

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.
F, S, OS-9 and SPECIAL CCF - $200.00, U - $395.00
OS-9 68K - $595.00

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DIET-TRAC Forecaster from S.E. Media -- An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.
F, S - $59.95, U - $89.95

GAMES

RAPIER - 6809 Chess Program from S.E. Media -- Requires FLEX, SK*DOS and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the strongest CHESS programs running on any microcomputer, *estimated USCF Rating 1600+ (better than most 'club' players at higher levels)*
F, S and CCF - $79.95

NEW

MS-DOS/FLEX Transfer Utilities For 68XXX and CoCo* OS-9 Systems. Now Read, Write, DIR, Dump and Explore FLEX & MS-DOS Disks. Supplied with a rich set of options to explore and transfer text type files from/to FLEX and MS-DOS disks. *CoCo OS-9 requires SDISK utilities & two floppy drives.
CCO $69.95 68XXX OS-9 $99.95

NOTE: Changes

1. Price increase for SCULPTOR, see advertising front of this catalog and other ad in this issue. Special price for 68 Micro Journal readers.
2. Lower price for BASIC09 TOOLS, see Utilities section.
3. New MS-DOS & FLEX to OS-9 Utilities, see above.

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

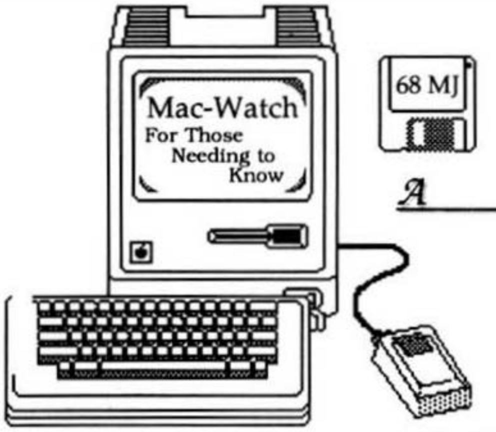
```
0B00CA 0C00 0046               CMP.B   #'F',D0         ASCII F
0B00CE 6E14      (B00E4        BGT.S   ERR
0B00D0 0400 0037               SUB.B   #$37,D0
0B00D4 E900             SHFT   ASL.B   #4,D0            000000B0
0B00D6 E980                    ASL.L   #4,D0            00000B00
0B00D8 5241                    ADD     #1,D1
0B00DA 0C01 0004               CMP.B   #4,D1
0B00DE 66CC      (B00AC        BNE.S   LOOP
0B00E0 A034                    DC      PCRLF
0B00E2 4E75                    RTS
                        *
0B00E4 49FA 01FA(B02E0 ERR     LEA     MESG(PC),A4
0B00E8 A035                    DC      PSTRNG
0B00EA 4E75                    RTS
                        * HELP MESSAGE PRINTS OUT ON COMMAND: MDUMP ?
                        *
0B00EC 49FA 0006(B00F4 HELP    LEA     HLPMSG(PC),A4
0B00F0 A035                    DC      PSTRNG
0B00F2 A01E                    DC      WARMST
                        *
0B00F4 4D44 554D 5020 HLPMSG   DC.B    "MDUMP requires no command line arguments.  It loads",$0D,$0A
0B0129 696E 2068 6967          DC.B    "in high memory, presently $0B0000 plus OFFSET.  It prompts",$0D,$0A
0B0165 666F 7220 6120          DC.B    "for a command.  N005E <cr> will cause it to dump a page ",$0D,$0A
0B019F 6F66 206D 656D          DC.B    "of memory, in this case from $005E00 to $005EFF.",$0D,$0A
0B01D1 636F 6D6D 616E          DC.B    "command F will cause the next page (forward) to be dumped",$0D,$0A
0B020C 616E 6420 4220          DC.B    "and B will cause the previous page (backward) to be dumped",$0D,$0A
0B0248 4561 6368 2070          DC.B    "Each page is dumped 16 bytes to a line with the ASCII",$0D,$0A
0B027F 7265 7072 6573          DC.B    "representation following the HEX.  Non-printable ASCII",$0D,$0A
0B02B7 6368 6172 6163          DC.B    "characters are represented as periods.",$0D,$0A,$04

0B02E0 4E4F 5420 4120 MESG     DC.B    "NOT A VALID ADDRESS",$04
0B02F4 434F 4D4D 414E PROMPT   DC.B    "COMMAND: ",$04
                               END     START
0 ERRORS DETECTED
```

```



                        ************************************************
                        * DISKFILE DUMP PROGRAM                        *
                        *                                              *
                        * COMMANDS:                                    *
                        * N XXXX NEXT PAGE TO BE DUMPED                *
                        * B BACK A SECTOR                              *
                        * F FORWARD A SECTOR                           *
                        ************************************************

                        * SK*DOS / 68K EQUATES FOR USER PROGRAMS

             0000A029   GETCH   EQU     $A029           Get input character with echo (7 bits)
             0000A023   GETNAM  EQU     $A023           Get filename into FCB
             0000A024   DEFEXT  EQU     $A024           Set default extension
             0000A005   FOPENR  EQU     $A005           Open file for read
             0000A001   FREAD   EQU     $A001           Read a byte
             0000A011   FSKIP   EQU     $A011           Next Sector
             0000A031   TOUPPR  EQU     $A031           Conv't char in D5 to Upper Case
             0000A02F   HEXIN   EQU     $A02F           Input hexadecimal number
             0000A03A   OUT2H   EQU     $A03A           Output 2 hex digits
             0000A03B   OUT4H   EQU     $A03B
             0000A03C   OUT8H   EQU     $A03C           Output 8 hex digits
             0000A034   PCRLF   EQU     $A034           Print CR/LF
             0000A036   PNSTRN  EQU     $A036           Print string (without CR/LF)
             0000A035   PSTRNG  EQU     $A035           Print CR/LF and string
             0000A033   PUTCH   EQU     $A033           Output character
             0000A000   VPOINT  EQU     $A000           Point to SK*DOS variable area
             0000A01E   WARMST  EQU     $A01E           Warm start

                        *
                        *
000000 A000             START   DC      VPOINT          GET POINTER
000002 284E                     MOVE.L  A6,A4           FCB POINTER
000004 A023                     DC      GETNAM          GET FILENAME
000006 65D0 0096(0009E          BCS     HELP
00000A 7801                     MOVE.L  #1,D4           CODE FOR .TXT
00000C A024                     DC      DEFEXT
00000E A005                     DC      FOPENR
000010 43FA 0032(00044          LEA     OLDTS(PC),A1
000014 328C 0003                MOVE.W  #$0003,(A1)     PRIME WITH BAD TS
000018 A001             LOOP    DC      FREAD
00001A 6526      (00042         BCS.S   ERROR
00001C 204C                      MOVE.L  A4,A0
00001E D1FC 0000 0060            ADD.L   #96,A0          POINT AT SECTOR INFO
000024 43FA 001E(00044           LEA     OLDTS(PC),A1
000028 3011                      MOVE.W  (A1),D0
00002A B050                      CMP.W   (A0),D0
00002C 6714      (00042          BEQ.S   ERROR           END OF FILE
00002E 3290                      MOVE.W  (A0),(A1)
000030 6114      (00046          BSR.S   OPAGE
000032 A029                      DC      GETCH
000034 A031                      DC      TOUPPR
000036 0C05 0045                 CMP.B   #'E',D5
00003A 6602      (0003E          BNE.S   CONTIN
00003C A01E                      DC      WARMST
00003E A011              CONTIN  DC      FSKIP
000040 6006      (00018          BRA     LOOP
000042 A01E              ERROR   DC      WARMST          ASSUME END OF FILE
000044                   OLDTS   DS.W    1
                         *
                         * ROUTINE TO OUTPUT A PAGE IN HEX AND ASCII
                         *
000046 A034              OPAGE   DC      PCRLF
000048 4240                      CLR.W   D0              LINE COUNTER
00004A A034                      DC      PCRLF
                         * LOOP FOR LINES
00004C 323C 000F         LLOOP   MOVE.W  #15,D1          COUNTER FOR CHARACTERS
000050 1800                      MOVE.B  D0,D4
000052 E904                      ASL.B   #4,D4           ADDRESS OF FIRST BYTE OF LINE ON PAGE
000054 A03A                      DC      OUT2H
000056 183C 0020                 MOVE.B  #$20,D4
00005A A033                      DC      PUTCH           SPACE
00005C A033                      DC      PUTCH           SECOND SPACE
                         * INSIDE LOOP FOR 16 CHARACTERS IN HEX
00005E 1818              CLOOP   MOVE.B  (A0)+,D4
000060 A03A                      DC      OUT2H           OUTPUT FIRST BYTE
000062 183C 0020                 MOVE.B  #$20,D4
000066 A033                      DC      PUTCH           SPACE
000068 57C9 FFF4(0005E           DBEQ    D1,CLOOP        CHARACTERS
                         * NOW DO ASCII CHARACTERS, "." FOR NON PRINTABLE
00006C 183C 0020                 MOVE.B  #$20,D4
000070 A033                      DC      PUTCH           EXTRA SPACE BEFORE ASCII
000072 91FC 0000 0010            SUB.L   #16,A0
000078 323C 000F                 MOVE.W  #15,D1          RELOAD COUNTER FOR CHARACTERS
00007C 1818              ALOOP   MOVE.B  (A0)+,D4
00007E 0244 007F                 AND     #$7F,D4         MASK OFF HI ORDER BIT
000082 0C04 0020                 CMP.B   #$20,D4         IS IT PRINTABLE?
000086 6C04      (0008C          BGE.S   AL1             IF YES
000088 183C 002E                 MOVE.B  #'.',D4         ELSE PRINT PERIOD
00008C A033              AL1     DC      PUTCH
00008E 57C9 FFEC(0007C           DBEQ    D1,ALOOP
000092 A034                      DC      PCRLF
000094 5240                      ADD     #1,D0
000096 0C00 0010                 CMP.B   #16,D0
00009A 66B0      (0004C          BNE     LLOOP           NEXT LINE
00009C 4E75                      RTS

00009E 49FA 0006(000A6 HELP      LEA     HLPMSG(PC),A4
0000A2 A035                      DC      PSTRNG
0000A4 A01E                      DC      WARMST

0000A6 5379 6E74 6178  HLPMSG    DC.B    "Syntax: DDUMP FILENAME   Defaults are work drive",$0D,$0A
0000D8 616E 6420 2E54            DC.B    "and .TXT extension.",$0D,$0A,$0A
0000EE 4444 554D 5020            DC.B    "DDUMP dumps a disk file to the terminal one sector",$0D,$0A
000122 6174 2061 2074            DC.B    "at a time.  The dump displays 16 bytes in HEX followed by",$0D,$0A
00015D 7468 6520 4153            DC.B    "the ASCII representation of the same 16 bytes.  Non-",$0D,$0A
000193 7072 696E 7461            DC.B    "printable characters are displayed as periods.",$0D,$0A
0001C3 4174 2074 6865            DC.B    "At the command prompt, F will cause the next (forward)",$0D,$0A
0001FB 7365 6374 6F72            DC.B    "sector to be displayed, and B (back) will display the",$0D,$0A
000232 7072 6576 696F            DC.B    "previous sector.  E will Exit the program as will",$0D,$0A
000265 4620 7768 656E            DC.B    "F when the last sector of the file has been reached.",$0D,$0A,$04

                                 END     START
 0 ERRORS DETECTED
```

```
                               * FIND UTILITY FOR SK*DOS /68K
                               * BY R. W. ANDERSON 1988
                               *
                               * EQUATES TO SK*DOS
                               *
                               * THE FOLLOWING WOULD NORMALLY BE IN SKEQUATE.TXT
                 0000A000      VPOINT  EQU     $A000
                 0000A038      OUT5D   EQU     $A038
                 00000001      FCBERR  EQU     1
                 0000A024      DEFEXT  EQU     $A024
                 0000A008      FCLOSE  EQU     $A008
                 0000A005      FOPENR  EQU     $A005
                 0000A006      FOPENW  EQU     $A006
                 0000A001      FREAD   EQU     $A001
                 0000A002      FWRITE  EQU     $A002
                 0000A023      GETNAM  EQU     $A023
                 0000A029      GETCH   EQU     $A029
                 0000A035      PSTRNG  EQU     $A035
                 0000A036      PNSTRN  EQU     $A036
                 0000A034      PCRLF   EQU     $A034
                 0000A037      PERROR  EQU     $A037
                 0000A033      PUTCH   EQU     $A033
                 0000A01E      WARMST  EQU     $A01E
                               *
000000                                 ORG     $0000
000000 6002         |00004 FIND    BRA.S   START           GOTO START
                               *
000002 0100                    VER     DC.W    $0100           VERSION NUMBER
                               *
000004 A034                    START   DC      PCRLF           START ON NEW LINE
000006 204E                            MOVE.L  A6,A0           SAVE POINTER
000008 284E                            MOVE.L  A6,A4           POINTER TO USER FCB
00000A A023                            DC      GETNAM          GET FILE SPEC
00000C 6404         |00012             BCC.S   NAMEOK          IF FILENAME OK
00000E 6000 0006|000E6                 BRA     HELP
                               * IF FILE SPEC WAS OK; DEFAULT TO .TXT
000012 183C 0001               NAMEOK  MOVE.B  #1,D4           DEFAULT EXTENSION
000016 A024                            DC      DEFEXT          DEFAULT EXTENSION
                               *
                               * NOW OPEN THE FILE
000018 A005                            DC      FOPENR          OPEN FOR READ
00001A 6660         |0007C             BNE.S   ERROR1          IF NOT ZERO
                               *
                               * MAIN LOOP TO READ LINES AND SEARCH FOR MATCH
                               *
00001C 49FA 01DE|001FC MAIN    LEA     PROMPT(PC),A4
000020 A035                            DC      PSTRNG          "INPUT SEARCH STRING"
000022 41FA 02EE|00312                 LEA     LINECT(PC),A0
000026 30BC 0000                       MOVE.W  #0,(A0)         CLEAR LINE COUNTER
00002A 6158         |00084             BSR.S   GETSTR          GET SEARCH STRING INTO BUFFER
00002C A034                            DC      PCRLF
                               *
00002E A000                    LINE    DC      VPOINT
000030 43FA 01E0|00212                 LEA     LINBUF(PC),A1
000034 284E                    LINE1   MOVE.L  A6,A4           POINT TO SYSFCB
000036 A001                            DC      FREAD           GO READ NEXT CHAR
000038 6636         |00070             BNE.S   ERROR
                               *
00003A 12C5                            MOVE.B  D5,(A1)+        PUT IN LINE BUFFER
00003C 0C05 0000                       CMP.B   #$0D,D5         IS IT LINEFEED
000040 66F2         |00034             BNE.S   LINE1           GET MORE
000042 137C 0004 FFFF                  MOVE.B  #$04,-1(A1)     CHANGE CR TO $04
000048 41FA 02C8|00312                 LEA     LINECT(PC),A0
00004C 5250                            ADD.W   #1,(A0)
00004E 6160         |000B0             BSR.S   MATCH
000050 4A00                            TST.B   D0
000052 6602         |00056             BNE.S   FOUND
000054 60D8         |0002E             BRA.S   LINE
000056 383A 028A|00312 FOUND   MOVE.W  LINECT(PC),D4
00005A 1A3C 00FF                       MOVE.B  #-1,D5          SET NON-ZERO
00005E A038                            DC      OUT5D
000060 183C 0020                       MOVE.B  #$20,D4
000064 A033                            DC      PUTCH
000066 49FA 01AA|00212                 LEA     LINBUF(PC),A4
00006A A036                            DC      PNSTRN
00006C A034                            DC      PCRLF
00006E 60BE         |0002E             BRA.S   LINE
                               *
000070 0C2C 0008 0001          ERROR   CMP.B   #8,FCBERR(A4)
000076 6604         |0007C             BNE.S   ERROR1          NOT END OF FILE
000078 6106         |00080 ERCLS   BSR.S   CLOSE
00007A A01E                            DC      WARMST
```

```
00007C A037              ERROR1  DC      PERROR
00007E 60F8      |00078          BRA.S   ERCLS
                         *
                         * SUBROUTINES
                         *
000080 A008              CLOSE   DC      FCLOSE              CLOSE FILE
000082 4E75                      RTS
                         *
                         * SUBROUTINE TO GET A STRING FROM TERMINAL INTO
                         * A BUFFER USING PC RELATIVE ADDRESSING
                         *
000084 41FA 020C|00292 GETSTR    LEA     BUFF(PC),A0         GET POINTER TO BUFFER
000088 A029              GET1    DC      GETCH
00008A 0C05 000D                 CMP.B   #$0D,D5
00008E 671A      |000AA          BEQ.S   EXIT
000090 0C05 0008                 CMP.B   #$08,D5
000094 6610      |000A6          BNE.S   GET2
000096 5388                      SUB.L   #1,A0
000098 183C 0020                 MOVE.B  #$20,D4
00009C A033                      DC      PUTCH
00009E 183C 0008                 MOVE.B  #$08,D4
0000A2 A033                      DC      PUTCH
0000A4 60E2      |00088          BRA.S   GET1
0000A6 10C5              GET2    MOVE.B  D5,(A0)+
0000A8 60DE      |00088          BRA.S   GET1
0000AA 10BC 0000         EXIT    MOVE.B  #0,(A0)             NULL TERMINATE STRING
0000AE 4E75                      RTS
                         *
                         * SUBROUTINE TO SEEK A MATCH BETWEEN A SEARCH STRING
                         * AND A LINE OF TEXT.  RETURNS ZERO IN D0 IF NOT FOUND,
                         * 1 IF FOUND.
                         *
0000B0 41FA 01E0|00292 MATCH     LEA     BUFF(PC),A0
0000B4 43FA 015C|00212           LEA     LINBUF(PC),A1
0000B8 2449                      MOVE.L  A1,A2               A2 KEEPS TRACK OF START OF MATCH
0000BA 1211              MATCH1  MOVE.B  (A1),D1             GET CHAR IN D1
0000BC B210                      CMP.B   (A0),D1
0000BE 6606      |000C6          BNE.S   MATCH2              IF NOT EQUAL MOVE DOWN LINE
0000C0 5288                      ADD.L   #1,A0               IF EQUAL COMPARE NEXT CHAR OF BUFF
0000C2 5289                      ADD.L   #1,A1               NEXT CHAR OF LINE
0000C4 6010      |000D6          BRA.S   MATCH3
0000C6 528A              MATCH2  ADD.L   #1,A2               IF NOT EQUAL START AGAIN
0000C8 224A                      MOVE.L  A2,A1               START AT NEXT CHAR IN LINE
0000CA 0C11 0004                 CMP.B   #$04,(A1)
0000CE 6712      |000E2          BEQ.S   NOTFND              GOT TO END OF LINE
0000D0 41FA 01C0|00292           LEA     BUFF(PC),A0
0000D4 60E4      |000BA          BRA.S   MATCH1              GO AROUND AGAIN.
0000D6 0C10 0000         MATCH3  CMP.B   #0,(A0)             MATCH
0000DA 66DE      |000BA          BNE.S   MATCH1
0000DC 103C 0001                 MOVE.B  #1,D0
0000E0 4E75                      RTS
0000E2 4200              NOTFND  CLR.B   D0
0000E4 4E75                      RTS
                         *
0000E6 49FA 0006|000EE HELP      LEA     HLPMSG(PC),A4
0000EA A035                      DC      PSTRNG
0000EC A01E                      DC      WARMST
                         *
0000EE 5379 6E74 6178    HLPMSG  DC.B    "Syntax:  FIND FILENAME    Defaults are work drive",$0D,$0A
000121 2020 2020 2020            DC.B    "             and .TXT extension.",$0D,$0A
000142 4649 4E44 2070            DC.B    "FIND prompts for a search string.  It will search the file",$0D,$0A
00017E 616E 6420 6C69            DC.B    "and list each line in the file that contains the string.",$0D,$0A
0001B8 5468 6520 6C69            DC.B    "The line number is included for convenience in editing",$0D,$0A
0001F0 7468 6520 6669            DC.B    "the file.",$0D,$0A,$04

0001FC 494E 5055 5420    PROMPT  DC.B    "INPUT SEARCH STRING: ",$04
000212                   LINBUF  DS.B    128
000292                   BUFF    DS.B    128
000312                   LINECT  DS.W    1
                                 END     FIND
 0 ERRORS DETECTED
```

EOF



The Macintosh™ Section

Reserved as

A place for your thoughts

And ours.......

Mac - Watch

A Review of Curator

An Art Management and Integration Program from Solutions International, Inc.

By: James E. Law
1806 Rock Bluff Road
Hixson, TN 37343

What Is The Problem to be Solved?

In the beginning, there was MacPaint and McWrite and the transfer of images between programs was simple. We understood that MacPaint produced bitmapped images in the PNTS format which when cut or copied, assumed the clipboard's PICT format for transfer. McWrite saved its documents in the WORD format or as plain ASCI code. Transfers of images took place effortlessly. That was all we needed to know.

The world has changed! The Macintosh environment may be the simplest now available, but even so, the explosion of second and third generation software has resulted in a diverse population of image formats, few of which are compatible with all applications. Postscript output drives postscript printers like the Apple LaserWriter. Encapsulated postscript contains the postscript code and a QuickDraw kernel that enables an approximation of the postscript image to be seen on the screen. The TIFF format is a high resolution bitmapped image commonly used with scanners. While TIFF is supposedly a standard format, I heard an industry insider claim that he personally knew of 47 versions of this format. A new "standard" format has just been introduced called RIFF. This format is reportedly much like TIFF, but requires only 30% of the storage space for an equivalent image. In addition to the large number of so called standard formats, there are a multitude of proprietary formats.

If we worked exclusively in one application, this would not be much of a problem. The real world is, however, that we use multiple applications using multiple output formats and with the obvious need to transfer images between programs. Programmers have taken some steps toward minimizing this problem. For example, SuperPaint documents can be saved in a PNTG (MacPaint) or PICT (Clipboard) format in addition to the proprietary format normally used by SuperPaint. As a result, SuperPaint images can be can figured for transfer into applications that only accept the PNTG format.

Such steps have not entirely solved the problem. For example, MacWrite does not understand Encapsulated PostScript and McDraw cannot read TIFF.

Then Along Came Curator

Curator allows you to have immediate access to the graphics you need regardless of what folder it is in. You do not have to set up a special art library. Curator can handle a variety of formats including PICT, MacPaint, TIFF, Postscript, Mac-Encapsulated Postscript, IBM-Encapsulated Postscript, plain postscript, and Glue. Then once you have found it, Curator converts the graphics from the format they were created in to the format you need.

Curator is provided in both application and desk accessory form. Both work essentially the same. Also provided is a small application called Curator's Assistant. This application automatically scans an entire floppy or hard disk and creates the Curator catalog with thumbnail sketches and key words. It is used when you first buy Curator to set up the initial catalog or to update the catalog later if you acquired a large quantity of additional graphics.

Getting Started

Drag the Curator and Curator Assistant icons to your disks and load the CuratorDA and you are ready to go. (Note that you may, but you do not have to load both the application and the DA version of Curator since both do the same thing.) Double click the Curator Assistant icon, respond to a dialog box, then work begins on constructing the Curator catalog. The manual warns you that this can take a long time. The catalog for my approximately 4 megabytes of graphics took less than 10 minutes.

Using Curator to Manage Art Collections

Curator allows you to search for and select graphics by name, thumbnail sketch, or key word.

Choosing the **Select by Name** option presents a typical dialog box used to select and open applications or documents. You can step through the various layers of folders then documents until you find what you want. You may select a different drive and eject a disk from this window. When you single click a document listed in the window, a 1" by 1-1/4" thumbnail sketch is displayed, the graphics format is indicated as is the application which created it. (Warning—These thumbnail sketches are so small that they do not effectively show some images. If you have a 8-1/2" by 11" image containing a number of small objects, as often occurs with commercial clip art, the thumbnail may not be recognizable. Of course, for bigger objects, the thumbnail sketches are helpful.) In any of your searches for graphics, you may limit the search to only a particular format or combination of formats. For example, you may limit your search to MacPaint and TIFF images thus excluding Mac EPS, IBM EPS, PICT, Postscript, and Glue.

The **Select by Thumbnail** option works essentially like the **Select by Name** option. Instead of a list of document names, you are presented with a window showing up to 10 thumbnails. You may scroll to see other thumnails if more than 10 documents are in the folder. Double clicking a thumbnailcauses that image to be displayed full size. Both the **Select by Name** and the **Select by Thumbnail** look at only one folder at a time. They cannot be used to do a global search of a volume.

The **Search by Name** option is different from the options listed above in that it searches the entire volume. (I think the choice of terminology in this program is unduly confusing. It is not intuitive why one option is called **Search by Name** and the other **Select by Name**.) You enter all or part of a document named and any matches will be listed. Also, you may search by format. For example, you could look for all TIFF graphics regardless of its name. A bar graph keeps you informed of the status (i.e., percent complete) of the search.

There seems to be a flaw in the design of this feature. A search will often yield multiple 'hits'. Suppose that you want to look at a full-sized image of each hit to determine which one you want. After you view the first, you close the image window and return to the list of hits for further exploration, right? Wrong! Your choices have been erased and you have to start over. This does not make sense.

Curator allows you to assign one or more keywords to each image on your floppy or hard disk. It also allows you to use the keywords which are sometimes preassigned by the publishers of clip art or the key words in PictureBase files. The lists of keywords can be viewed and changed at any time. This feature can provide tremendous power in finding exactly the desired image quickly. It only works, however, if you are disciplined enough to assign meaningful keywords for each image.

Curator creates an index of all keywords on the entire floppy or hard disk. When you select **Search by Keyword**, a scrolling list of available key words is presented. You may then select on or more of these for inclusion in your search criteria.

The final search option provided by Curator is called **Browse**. This option allows you to step through the images of a selected folder to view a small portion of the image, assign or change keywords, or select and open documents. The image is viewed at full size through a 5" by 2" window. Moving form one image to another takes 5 or 6 seconds on my system.

Using Selected Images

When you find the image you want, you open it by clicking the appropriate button. The image is then viewed in a window which can be zoomed to the size of the Macintosh window. Scroll bars are provided for precise (but slow) positioning of the image. No grabber hand is available for faster positioning.

The image may be viewed, cut or copied, printed, or saved in a different format. The cutting and copying is done by choosing **Select All** from the **Edit Menu** or by using the selection rectangle. The printing involves use of standard **Page Setup** and **Print** dialog boxes. You may also view or change the key words associated with the displayed image or may view the list of all key words in the catalog.

Converting Image Formats

The most useful and straightforward feature of Curator is the ability to convert images from one format to another. This is accomplished merely by checking the desired new format, then using the SAVE AS. . . command. The supported conversions are indicated by dimming the box for those that are not supported (e.g., MacPaint to Mac EPS.)

The Curator Manual

The manual says, "You may not need to read this manual at all!" That may be true, but only if you are the one that wrote Curator. This program is about as intuitive as driving a car in Boston. After a lot of driving and many wrong turns, maybe you will get there. The manual usually (but not always) does an adequate job of explaining the product. The section of **Browsing** is confusing. It seems to contain an irrelevant illustration (i.e., it shows the wrong window) and leaves out a critical step.

The Bottom Line

Solution International, Inc., had good intentions in developing Curator. This program addresses a problem which needs to be solved (i.e., the multiplicity of graphics formats.) The implementation, however, is in my opinion, not up to this company's usual standards. The interface of the art management part of the program needs to be simplified. Also, the manual needs to be clarified, especially as it describes the **Browsing** function.

The most important function of Curator, however, is image conversion and this function is simple and powerful. It does what it claims it will do.

If you need the ability to convert graphics from one format to another, buy Curator. If all you need is an art management program, wait until it is refined.

EOF

Pascal

A Tutorial

By: Robert D. Reimiller
Certified Software Corp
616 Camino Caballo
Nipomo, CA 9 444
805 929-1359

This month we are going to look at variable allocation, but before we do, let's take a look at "structured" constants. Structured constants refer to constant arrays and/or records. These tend to straddle the line between constants and variables in all three languages. Pascal and Modula-2K both have true structured constants. C uses an equally useful mechanism of initialized variables.

In Pascal, a structured constant in defined in one of three ways :

```
const
  abc = local <type> <constant-list> ;
  abc = entry <type> <constant-list> ;
  abc = external <type> ;
```

A local structured constant is only known within the module or procedure where it is declared. An entry structured constant in known within the module where it is defined, and any other modules that declare the same name as external. Entry/external structured constants cannot be defined within a procedure. The constant list consists of the constant value in the case of simple types, such as integer, boolean, string, or constants within in parenthesis in the case of arrays or records (one level of parenthesis for each level of record or array). For instance, all of these are valid :

```
const
  abc = local integer 0 ;
  def = entry string 'this is a test' ;
  ghi = local array [1..5] of integer
(10,100,1000,50,70) ;
  jkl = entry record
        a : integer ;
        b : string ;
        c : array [1..3,1..4] of integer
      end
(25,'test',((5,10,15,20),(10,20,30,40),
(20,40,60,80))) ;
```

In Modula-2K the distinction if the constant is local, entry, or external is determined by whether or not the constant is mentioned in the definition module (indicating export (entry)), or whether it is in an import list (indicating import (external)). If it is in neither of these, then it is a local structured constant. As in the Pascal, if it is external (imported), then the constant-list is not included. Also, the constant list is not specified when defining an exported constant in the definition module. So for Modula-2K :

```
DEFINITION MODULE

                                  (local)
const abc = <type>                (entry)
                                  (external)

IMPLEMENTATION MODULE

const abc = <type> <list>         (local)
const abc = <type> <list>         (global)

from <module> import abc          (external)
```

In Pascal and Modula-2K these constants can be indexed, and fields accessed just like a structured variable.

In C a totally different concept is used, that of the initialized variable. Whether or not the variable is local to the module follows the same rules as normal variables. The method of definition is almost the same as Pascal and Modula-2K :

```
int i = 0 ;
int x[4] = {10, 20, 30, 40} ;
```

The implementation of initialized variables varies depending on the operating system environment, but in general the data is copied from the memory module to the data area before execution of the C program, resulting in two copies of the data in memory when the program starts.

In the Pascal and Modula-2K implementation, the data stays in the memory module, since it is not to be modified. Wanting to have an actual initialized variable (versus a set of constants, such as a look-up table) is rare, but when required can be done with just a simple assignment, resulting in essentially the same situation as in C's initialized variables.

Now onto real variables. The default allocation method in Pascal is on the stack. This applies to any variables declared globally, or local to a procedure. This is the only allocation method in standard pascal. In a modular environment it is vital that the global stack variables be declared identically in all modules, this is assured by declaring them in an include file. Modula-2K and C do not have a global stack, but have a local stack for local procedure variables.

"Register" and "FPR" variables are also allowed. Register variables are stored in the CPU registers (D4-D7) and up to 4 are allowed per procedure. Likewise, FPR variables are stored in the 68881 registers (FP2-FP7) and up to 6 are allowed per procedure. Both of these types are available in Modula-2K as well. C allows register variables, how many are allowed per procedure, and whether or not they are stored in the CPU or FPU registers would naturally depend on the specific compiler used.

```
        PASCAL                      MODULA-2K

i : integer register    i [register] : integer

                    C

            register int i;
```

The "Varib" type is unique to Pascal, since it's default type is stack storage, and the other two languages default to the equivalent of "Varib". "Varib" simply means that the storage location of the variable (relative to a base register) is determined by the linker, not the compiler. In Pascal "Varib" may store up to 32K bytes of data. In Modula-2K and at least one C compiler, up to 64K bytes of data is allowed. In Pascal, Modula-2K and at least one C compiler, another section called "Remote" is allowed, which is like "Varib", but uses 32 bit address and allows up to 2 giga bytes of storage.

There are also variants of these type depending on whether you want the variable to be known outside of this module. If the variable is declared inside a procedure, then there are no possible options, it is not known outside of that procedure (normal scope rules).

In Pascal if there is no special modifier after the "Varib" or "Remote" then it is not known outside of the module or procedure where it is declared. To make the variable known outside of this module, it is followed by the word "Entry", and to access this variable in another module, by the word "External".

```
i : integer varib {local}
i : integer varib entry {global}
i : integer varib external {global, defined
                            elsewhere}
i : integer remote entry {remote global}
```

In Modula-2K the distinction is defined by the appearance in the definition module or an import list :

DEFINITION MODULE

```
                                 {local}
i : integer                      {global}
                                 {global, defined
                                  elsewhere}
i [remote] : integer             {global remote}
```

IMPLEMENTATION MODULE

```
i : integer                      {local}
i : integer                      {global}
from <module> import i           {global, defined
                                  elsewhere}
i [remote] : integer             {global remote}
```

In C a variable declared outside of a procedure without any modification of the declaration type is essentially a globally known variable (the same as varib entry in Pascal), and can be referenced in other modules by using the keyword "extern" (the same as varib external in Pascal). To make a variable that is global to a module, but not known outside the module, the keyword "static" is used.

```
static int i {local}
int i {global}
extern int i {global, defined elsewhere}
remote int i {global remote}
```

One very useful mode found in Pascal and Modula-2K is the absolute addressing mode. This allows a variable to be located at a specific location in the memory map, which is mostly used for I/O ports. As examples :

PASCAL

```
i : integer at $ff104 ; {I/O port}
a : array [1..24,1..80] of char at $e0000 ;
                              {memory mapped screen}
```

MODULA-2K

```
i [$ff104] : integer ; {I/O port}
a [$e0000] : array [1..24,1..80] of char ;
                         {memory mapped screen}
```

In C, this facility is not available, and pointers are often used to access absolute addressing by assigning the address to the pointer variable. This method unfortunately generates more code, but there is a way of accessing absolute addresses using the same code as Pascal and Modula-2K by using a "caste" :

```
#define i * (integer *) 0xff104
```

Unfortunately this does not seem to work for arrays or records, and so you are back to using pointers or defining each individual field of a record at a specific address.

Next time we will look at operators.

EOF.

A condensed outline of utilites that many have asked for. They are now available from

South East Media (see catalog.)

GCS FILE TRANSFER UTILITIES

GCS FILE TRANSFER UTILITIES are a group of programs to transfer files on MS-DOS (PC) and FLEX format floppy disks to and from OS-9.

MS-DOS text files can be transferred, but not binary files. Remember that Intel and Motorola operation codes are incompatible. Both text and binary FLEX files can be transferred. Binary FLEX files are not directly executable by OS-9.

The use and syntax of these utilities is standard OS-9. Usually, the default values for number of sides, sectors, etc., will suffice. A number of options are available so that other standard MS-DOS and FLEX formats, as well as some other formats, can be read and written.

NOTES - General:

Options can be displayed by entering: <prognam> -?.

Optional <devnam> for the device where the PC or FLEX disk is located must begin with a '/', (for example: /D1).

In the program descriptions which follow, the default values are assumed.

Certain features of OS-9 and MS-DOS file handling such as complete hierarchical handling, have not been included in these programs.

First level MS-DOS sub-directories can be read by PCDIR, PCREAD, PCDELETE and PCRENAME.

NOTES - 40 and 80 Track Drives:

MS-DOS files are normally written on disks formatted for 40 tracks on a 40 track drive. This is generally true for FLEX format disks. Assuming that your drives are properly aligned, you will be able to read MS-DOS and FLEX disks on either 40 or 80 track drives. The defaults are set for 80 track drives; use the -4 option for a 40 track drive. (For EXPLORE, the sense is reversed - the default is 40 track and the option is -8). Writing files to MS-DOS or FLEX disks should be done using a 40 track drive, if the disks were originally formatted on 40 track drives. If an eighty track drive is used to write to disks formatted on a 40 track drive, it is unlikely that any files so written will be read reliably on a PC or FLEX system. See the GMX Micro-20 Manual Addenda for OS-9 (page 3-2).

PCDIR

This program displays the directory of a PC format disk in /D0.

The program will read the root directory or a directory in the root directory. The path for a directory in the root directory is: <dir>.

Usage

Function: display pc format disk file directory

Syntax: pcdir [<opts>] [<devnam>] [<opts>]

Options: -e extended directory listing -f display disk format -d single density -s single sided -4 40 track drive -c=<num> sectors per cluster -n=<num> sectors per track -t=<num> tracks per surface

-? usage

Default - options: 80 track drive, double density, double sided, 2 sectors per cluster, 9 sectors per track, 40 track disk, 512 bytes per sector

- devnam: /d0

PCDUMP

This program displays individual sectors of a PC format disk in /D0.

The user is prompted for track, sector and side.

After a sector has been displayed, the user is prompted for:

N - next sector P - previous sector R - repeat current sector S - start over Q - quit

Output is to <stdout>. Redirection can be used for output to a file or printer.

Usage

Function: display pc format disk files by specified track, sector and side

Syntax: pcdump [<opts>] [<devnam>] [<opts>]

Options: -d single density -s single sided -4 40 track drive -c=<num> sectors per cluster -n=<num> sectors per track -t=<num> tracks per surface

-? usage

Default - options: 80 track drive, double density, double sided, 2 sectors per cluster, 9 sectors per track, 40 track disk, 512 bytes per sector

- devnam: /d0

PCREAD

This program reads a file contained in a PC format disk in /D0.

The file must either be in the root directory or a first level sub-directory.

The path for a file in the first level sub-directory is:

<dir><file>.

Line feeds are stripped from output. Use the <-l> option to retain the line feeds.

Output is to <stdout>. Redirection can be used for output to a file or printer.

Usage

Function: read pc format disk files

Syntax: pcread [<opts>[[devnam] <path> [devnam] [<opts>]

Options: -l pass line feeds -d single density -s single sided -4 40 track drive -c=<num> sectors per cluster -n=<num> sectors per track -t=<num> tracks per surface

-? usage

Default - options: 80 track drive, double density, double sided, 2 sectors per cluster, 9 sectors per track, 40 track disk, 512 bytes per sector, strip line feeds

- devnam: /d0

PCWRITE

This program writes a OS-9 file to a file on a PC format disk in /D0.

The file may be written only into the root directory.

If no destination filename (dstfile) is specified in the command line, the OS-9 file (srcfile) is used as the destination filename).

If the OS-9 filename is longer than 8 characters, the destination filename written into the PC disk will be truncated. If the desti- nation filename is included in the command line, a more suitable filename and extension can be used (8 characters for filename - 3 characters for extension).

NOTE: Please observe the caution on writing to 40 and 80 track drives. For reliable results, 40 track disks should only be written with 40 track drives. Usage

Function: write pc format disk files

Syntax: pcwrite [<opts>] [<devnam>] <srcfile> [<dstfile>] [<devnam>] [<opts>]

Options: -l don't add line feeds -d single density -s single sided -4 40 track drive -c=<num> sectors per cluster -n=<num> sectors per track -t=<num> tracks per surface

-? usage

Default - options: 80 track drive, double density, double sided, 2 sectors per cluster, 9 sectors per track, 40 track disk, 512 bytes per sector, add line feeds

- devnam: /d0

PCDELETE

This program deletes a file on pc format disk. Please be careful with this program! Once a file is deleted, it cannot be restored because the cluster chain for the file is cleared from the file allocation tables. This is done so that the space on the disk can be reused to write other files.

The prompt "pcdelete: Deleting: <filename> - Are you sure (Y/N) ?" provides an opportunity to abort the delete.

The path for a file in a first level sub-directory is:

<dir><file>.

Usage

Function: delete pc format disk file

Syntax: pcdelete [<opts>] [devnam] <filename> [devnam] [<opts>]

Options:

-d single density -s single sided -4 40 track drive -c=<num> sectors per cluster -n=<num> sectors per track -t=<num> tracks per surface

-? usage

Default - options: 80 track drive, double density, double sided, 2 sectors per cluster, 9 sectors per track, 40 track disk, 512 bytes per sector

- devnam: /d0

PCRENAME

This program renames a file contained on a pc format disk.

The path for a file in a first level sub-directory is:

<dir><file>

Usage

Function: rename a pc format disk file

Syntax: pcrename [<opts>] [<devnam>] <old file name> <new file name>

Options:

-d single density -s single sided -4 40 track drive -c=<num> sectors per cluster -n=<num> sectors per track -t=<num> tracks per surface

-? usage

Default - options: 80 track drive, double density, double sided, 2 sectors per cluster, 9 sectors per track, 40 track disk, 512 bytes per sector

- devnam: /d0

PCFORMAT

This program formats a floppy disk with pc format. While a disk can be formatted on either a 40 or 80 track drive, the disk is actually formatted as 40 track. If a disk is formatted on a 80 track drive and is always written on a 80 track drive, the disk can be read on either a 40 or 80 track drive.

For reliable results, format and write on 40 track drives, since pc systems are normally 40 track.

It is not normally necessary to verify formatting given the quality of most current day floppy disks. The -q option verifies the first sector on each track. The -v option verifies all sectors on all tracks. Verifying does slow down the format process.

Usage

Function: format pc disk

Syntax: pcformat [<opts>] [<devnam>] [<opts>]

Options:

-l prompts for volume label -q quick verify -s single sided -v verify -4 40 track drive

-? usage

Defaults - options: 80 track drive, double density, double sided, 40 track disk, 9 sectors per track, 2 sectors per cluster, 512 bytes per sector, no volume label, no verify

devnam: /d0

FLEXDIR

This program displays the directory of a FLEX format disk in /D0.

Usage

Function: display FLEX format disk file directory

Syntax: flexdir [<opts>] [<devnam>] [<opts>]

Options: -e extended directory listing -s double sided -n=<num> sectors per track

-? usage

Default - options: single density, single sided, 0/80 track disk, 10 sectors per track, 0/80 track drive - devnam: /d0

FLEXDUMP

This program displays individual sectors of a FLEX format disk in /D0.

The user is prompted for track, sector and side.

After a sector has been displayed, the user is prompted for:

N - next sector P - previous sector S - start over Q - quit

Output is to <stdout>. Redirection can be used for output to a file or printer.

Usage

Function: display specified FLEX format disk file sector

Syntax: flexdump [<opts>] [<devnam>] <file> [<devnam>] [<opts>]

Options: -d double density -s double sided -4 40 track drive -n=<num> sectors per track -t=<num> tracks per surface

-? usage

Default - options: 80 track drive, single density, single sided, 40 track disk, 10 sectors per track,

- devnam: /d0

FLEXREAD

This program reads a file contained in a FLEX format disk in /D0.

Use the -b option to transfer binary files directly to a OS-9 file.

If formatted binary output is desired for screen or printer output; also use the -f option.

Usage

Function: read FLEX format disk files

Syntax: flexread [<opts>][<devnam>] <file> [<devnam>][<opts>]

Options: -b binary file -f format binary output -d double density -s double sided -4 40 track drive - n=<num> sectors per track -t=<num> tracks per surface -? usage

Default - options: 80 track drive,single density, single sided, 40 track disk, 10 sectors per track, text file

- devnam: /d0

FLEXWRITE

This program writes a OS-9 file to a file on a FLEX format disk in /D0.

If no destination filename (dstfile) is specified in the command line, the OS-9 file (srcfile) is used as the destination filename.

If the OS-9 filename is longer than 8 characters, the filename (dstfile) written into the FLEX disk will be truncated. If the desti- nation filename is included in the command line, a more suitable filename and extension can be used (8 characters for file - 3 characters for extension).

Embedded tab characters (09) in the source file file are replaced by a tab character followed by a tab count character. The default tab count character default if 4. This can be changed use of -e option.

Strings of space characters ($20) in the source file are replaced by a tab character ($09) followed by a space count character. This creates normal FLEX space compression.

NOTE: Please observe the caution on writing to 40 and 80 track drives. For reliable results, 40 track disks should only be written with 40 track drives.

Usage

Function: write FLEX format disk files

Syntax: flexwrite [<opts>] [<devnam>] <srcfile> [<dstfile>] [<devnam>] [<opts>]

Options: -b binary file -e=<num> spaces per tab -d double density -s double sided -4 40 track drive -n=<num> sectors per track -? usage

Default - options: text file, 4 spaces per tab, 80 track drive, single density, single sided, 40 track disk, 10 sectors per track, 256 bytes per sector

- devnam: /d0

EXPLORE

This program can be used to display files on a variety of disk file formats.

It should work on disks which have been formatted using one of the IBM formatting algorithms with a controller using a WD1770, WD1772, WD1773 or WD179x chip.

Usage

Function: display disk files by specified track, sector and side

Syntax: explore [<opts>] [<devnam>] [<opts>]

Options: -g GMX Micro-20 configuration -b 512 bytes per sector -d double density - 16 sectors per track -s double sided -q 80 track density disk -8 80 track drive -n=<num> sectors per track -t track 0 always single density -m strip most significent bit of data byte -? usage

Default - options: 40 track drive, single density, single sided, 10 sectors per track, all tracks same density, 40 track disk, 256 bytes per sector, pass most significent bit of data byte

- devnam: /d0

INSTALLATION and TUTORIAL

This is a simple tutorial for the GCS File Transfer Utilities package.

First copy the utilities into your CMDS directory using the "load_util " procedure file.

Enter the following command line:

pcformat -? This will display the syntax and options for this command. (-? has the same function for other commands.)

Insert an unformatted disk into D0 device - it is assumed that this is an 80 track drive. If it is a 40 track drive, be sure to use the -4 option.

If you want to use another device; use /D1, for example, on the command line.

Enter the following command line:

pcformat

The program will display the device on which the disk to be formatted is mounted (/D0 for default). You will be asked if you wish to continue, enter: y . Formatting will proceed; at the end of a successful format, a summary of good and bad sectors will appear. If formatting should abort, an explanatory message will appear.

Use the CHD command to set up the OS-9 source/ destination directory.

Now try writing a file to the MS-DOS disk.

Enter the following command line:

pcwrite file1

(file1 = any text file in the current data directory)

Note that the file name may be truncated if it is longer than 8 characters since that is the maximum length for a MS-DOS file name.

Check that this has taken place by entering the following command line:

pcdir 'file1' will be the only file shown in the directory.

Read this file disk by entering the following command line:

pcread file1

This will display 'file1' on your terminal. Note that extraneous line feeds have been stripped from the display. If you want to keep line feeds, use the -l option. On the terminal, this option should result in double spacing.

Enter the following command line:

pcread file1 >-another_name This command line will write 'file1' into the current data directory. Don't forget the '-' after the '>' and use a different file name.

Check that this has taken place by entering the following command line:

list another_name

Usually the default values will work for most MS-DOS disks, since this is the most common current format. The 'pcdir' command can read several standard MS-DOS formats. The -f option for this command will indicate which format is used so that you can use appropriate options with the other commands if necessary.

Enter the following command line:

pcdir -? This will display the syntax and options for this command.

Enter the following command line:

pcdir

This will display the files in the directory of the MS-DOS disk - just showing the names of the files.

Enter the following command line:

pcdir -e This will display an extended listing of the directory.

Enter the following command line:

pcdir -f This will display the format parameters of the MS-DOS disk if it is one of the standard formats.

Using the preceeding command lines, you will receive an idea of the use of this command. Options can be combined as needed. They can go before or after filenames. Likewise device name.

As you can see these are a very complete set of utilities. While there have
been other utilities available to our readers. This seems to be the most complete set we have seen so far.

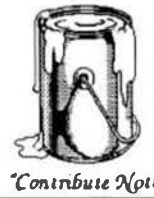
Available for 68XXX OS-9 and CoCo OS-9 from S.E. MEDIA.

Price: $99.00

EOF

FOR THOSE WHO NEED TO KNOW **68 MICRO JOURNAL™**

Bit-Bucket



By: All of us

"Contribute Nothing · Expect Nothing", DMW '86

microware

MICROWARE SYSTEMS CORPORATION
1900 N.W. 114th Street
Des Moines, Iowa 50322

Phone 515-224-1929
Telex 910-520-2535
FAX 515-224-1352

FOR MORE INFORMATION CONTACT:
Mr. Andy Ball
Vice President, Marketing
Microware Systems Corporation
1900 NW 114th Street
Des Moines, Iowa 50322
515-224-1929

MICROWARE ANNOUNCES THE RELEASE OF DEVELOPMENT PAKS FOR THE MOTOROLA VME 133A, 134 & 135

DES MOINES, Iowa, -- Microware Systems Corporation announces the release of three new OS-9 Operating System Development Paks for the Motorola VME 133A, 134 and 135 MC68020 monoboard microcomputers. OS-9 Development Paks provide a complete C Language development environment including a Kernighan & Ritchie compatible C compiler, assembler, linker, debugger, screen editor and over 50 other popular utility commands. Microware's OS-9 real-time operating system is used by major manufacturers world wide in a variety of applications ranging from embedded process control systems, multi-user development systems and consumer electronics.

These OS-9 Development Paks have been customized to support a variety of controllers for floppy and hard disks, magnetic tape, and serial and parallel I/O. Controllers supported by Microware's Development Paks include the MVME 319, 320, 335, 350 and 050.

The Microware MVME 133A Development Pak includes complete device driver or trap handler support for the on-board MC68881 FPU, RS-232 serial ports and 20Mhz clock. These features make this microcomputer a popular choice for time-critical or numerically intensive industrial applications.

The MVME 134 monoboard microcomputer includes a MC68851 Paged Memory Management Unit (PMMU). OS-9 memory protection support for this device and 4 Mbytes on-board RAM make the MVME 134 Development Pak an ideal engine for large multi-user development systems.

A significant feature of the MVME 135 is a VSB private memory bus. The VSB bus optimizes VME bandwidth utilization by minimizing CPU accesses to the VME-bus. OS-9 support for this private memory bus makes the MVME 135 especially useful for multi-processor systems or systems incorporating high-speed DMA controllers.

The three OS-9 Development Paks are immediately available from Microware and authorised Microware distributors. Please contact Microware for pricing information.

The OS-9 Operating system is a real-time, multi-user and multi-tasking system for computers based on the Motorola family of 68xxx processors. OS-9 is compact, ROMable and provides a UNIX-style environment for application software. Since its introduction in 1983, OS-9/68000 has been licensed to over 350 original equipment manufacturers (OEMs) world-wide for use in a variety of industrial, scientific and consumer products.

Computer Systems Consultants, Inc.
1454 Latta Lane
Conyers, GA 30207

Don Williams, Editor
68 Micro Journal
5900 Cassandra Smith
Hixson, TN 37343

Dear Don:

As Leo Taylor mentioned in his letter to the editor in 68 Micro, the current version of Super Sleuth for the 68010 was designed to disassemble 68000, 6800B, and 68010 machine code, which it does. This current version has no current known bugs. I have used it for some substantial disassemblies, and customers have reported successfully using it for disassembling several large programs, including monitors and operating systems.

The OS9/68000 version of Super Sleuth does not currently process the OS9/68000 header and system calls; however, it will process the remainder of the machine code properly. The table of system calls in Super Sleuth is not currently used, but is for use in the future version. Both the format of the OS9/68000 file header and the exact format of the OS9/68000 system calls are badly-documented, making this task much more difficult than it was under OS9/6809.

This system-specific information will be processed in a future version of Super Sleuth which will also process the UNIX V header and system calls.

Incidentally, Leo Taylor's description of the UNIX file system is not accurate. It is substantially more robust than his description would indicate. There are also two major current versions of the file system, as originated by Berkeley and as originated by AT&T. Those who are interested may read one of the many books describing the internals of the UNIX operating system.

Thank You,

Bud

E. M. (Bud) Pass

MICRONICS
RESEARCH CORP.

Microcomputers - Hardware and Software
GIMIX® Sales, Service and Support

Dear Don,

How time flies! Hadn't realised it's been so long since I began writing about logic functions in BASIC, as I've been so busy with my 68000 version of RBASIC, plus a design project which suddenly cropped up. So let's see if I can pick up the thread of where I left off.

I was emphasising just how careful we have to be when modifying logic expressions which include ELSE as part of their evaluation. Now for a much different example, which begins by looking like a mere extension of my earlier discussion ... but just wait till the punch-line!! Let's look at the following :

```
  100  IF X% < 3 GOTO 120
  110  IF Y% = 9 GOTO 130
  120  IF X% < 2 THEN Z% = 3
  130 rest of program
```

Note that we can arrive at Line-120 by two different routes from Line-100 (i) directly if (X%<3) or (ii) failing that, and falling through to Line-110, if (Y% <> 9). So let's try combining Lines 100 and 110 into

```
  100  IF X% > 2 AND Y% = 9 GOTO 130
  120  IF X% < 2 THEN Z% = 3
  130  rest of program
```

This is permissible because the program will arrive at Line-110 in the original program only if (X% > 2), and then only if (Y% = 9) will it bypass Line-120 and GOTO 130. We haven't changed the intent of the original at all, as the program will still arrive at Line-120 if (X% < 3) or if (Y% <> 9). Aha! So how about now combining Lines 100 and 120 so

```
    100  IF (X% < 3 OR Y% <> 9) AND X% < 2 THEN Z% = 3
    130  rest of program
```

I hope you'll remember from my 'Logically Speaking' series how to negate the two conditionals of the old Line-100 to produce the OR-expression inside the parens in our new Line-100. If not, I'll summarise it as "The opposite of (X% > 2) is (X% < 3); the opposite of AND is OR; and the opposite of (Y% = 9) is (Y% <> 9). **So our old program will reach Line-120 under this combination, ie, (X% < 3 OR Y% <> 9), AND then if (X% < 2)** Z% will get set to 3. Don't be tempted to omit the parens around the pair of OR-coupled terms, otherwise BASIC will read the line as

```
   100  IF X% < 3 OR (Y% <> 9 AND X% < 2) THEN Z% = 3.
```

Remember that AND equates to multiplication, and OR to addition, so the two terms coupled by AND will have a higher mathematical priority than OR, and BASIC will see a set of implied parens around the AND-coupled-terms.

Normally this would be about as far as we could go with reducing the complexity of the original program, but now we have a whole army of Boolean Algebra experts out there as a result of my 'Logically Speaking' tutorial series, and I doubt that they'll be content to let things rest there. As a first step, they'd expand the logic terms into

```
    (X% < 2).(X% < 3) + (X% < 2).(Y% <> 9)
```

and then they'd observe that the first product-term can be reduced to a simple (X% < 2). After all, if something is **less than** 3 and at the same time is **less than 2**, the condition can obviously be met by the **simple condition less than 2.** So now we have

```
    (X% < 2) + (X% < 2).(Y% <> 9)
```

And, as a final step, we'll apply Rule 5 of the Laws of Boolean Algebra (see page 23 of the September 1987 issue of 68MJ), which states that 'where a term in **its entirety** forms part of a larger term, the larger term can disappear.' This, of course - where (X% < 2) corresponds to 'a' and (Y% <> 9) to 'b' - reduces us to a mere

```
    (X% < 2)
```

so we can now simplify our program to

```
    100  IF X% < 2 THEN Z% = 3
    130  rest of program
```

All of which, though it's not intuitively obvious in our original program, tells us that the original Lines 100 and 110 are redundant, their only function being to act as 'program-complicators' and 'time-wasters'.

I hope this little example will serve to encourage others to apply Boolean Algebra to their BASIC programs, as well as to circuit design!!

And now, before signing off, I have to confess to having fallen into a trap of my own making. I should have known better, but a reader, Dexter S. French, pointed out that in my number-base conversion program (in XBASIC XPLANATIONS) I said that the following

```
    40  K%=ASC(MID$(N$,I%,1))-48: IF K%>9 THEN K%=K%-7
```

could be shortened, by using logic expressions, into

```
    40  K%=ASC(MID$(N$,I%,1))-48 +7*(K%>9)
```

which, I'm afraid, **just ain't true!** This comes of not checking out one's own program - - no matter how straightforward it looks. The first Line-40 is OK, because by the time the conditional (K%>9) is examined, it has already been calculated in the first part of the line.

Unlike pocket-calculators, however, BASIC doesn't update the value of a variable **until it's completed its whole chain of calculations** in a set of Math-registers, so the second Line-40, in the latter portion, is basing its logic decision on a previous value of K%, and **not** on its value at the point where it's just subtracted 48. The math operation is incomplete at this point, as there's still something to be added to the continuing evaluation of K%, so K% is not yet due to be updated.

Although I'm thoroughly aware of the fact that one shouldn't include logic-functions where the value of the variable has not yet been determined, I slipped up, and this one sort of got by me. Sorry about that! Please amend your copies of my 'XBASIC XPLANATIONS' accordingly.

I'll be in touch.

Don Williams,
68 Micro Journal,
5900 Cassandra Smith Road,
Hixson, TN 37343

Sincerely,

[illegible]

R. Jones
President

GENERAL MICRO SYSTEMS
INCORPORATED

25MHz 32-BIT VME MULTIPROCESSING CPU BOARD WITH NO-WAIT-STATES NOW SHIPPING

Scott Bowman
(714) 625-5475

Montclair, California—The first commercially available 25MHz 68020 CPU board with 1Mbyte of dual ported, zero-wait-state RAM is now shipping for VMEbus applications. General Micro Systems designed the GMSV07-25MHz CPU for bus-traffic intensive applications by optimizing the onboard logic for high-speed operation. The V07 offers a direct upgrade path from 16MHz to 25MHz and faster by incorporating the latest logic devices and by using efficient parallel logic design.

The V07 was designed with innovative multiprocessing capabilities for interprocessor communication and flexible system architecture. Multiple location monitors allow prioritized bus interrupt generation. Protected local memory masks the V07's local memory from unwanted writes yet permits dual porting. Multiple CPUs can be booted without ROM via the Bus Master Boot feature. The GMS V07 SAM™ local bus expansion offloads the VMEbus for memory access and I/O functions.

The 25MHz -performance of the V07 means computational tasks can be accomplished with fewer CPUs, resulting in the simplified system architecture necessary for practical multiprocessing systems. A feature contributing to this streamlined multiple-CPU environment is distributed interrupt handling which dynamically allocates interrupt handling functions between processors on the VMEbus. The V07 can both originate and service interrupts to the VMEbus. Since General Micro Systems' SAM™ mezzanine modules transform a V07 into an intelligent special purpose I/O processor, an entire system can be built from identical foundation boards and software, easing development and improving system reliability.

The high throughput of the GMSV07-25MHz underscores the need for parallel data transfer paths. A unique option of the V07 is a SAM™-VSB Master/Slave module that yields two true subsystem buses for direct CPU to CPU communication.

In addition to high speed and multiprocessing features the GMSV07-25MHz improves the logistics of operating multiple CPU systems. The V07's unique Bus Master Boot requires only one boot PROM(s) for the entire multiple CPU system. PROM sockets become available for up to an additional 512K of RAM per board. Upon boot-up the V07 master installs the code into the memory of slave CPUs and starts them. Code executes faster and, since changes in boot code need only be done once, system software can thus be debugged more easily.

Thus each board is independent of boot code and is interchangeable within the system. A customer need only put in the PROM(s) for the system master and boot it up. Since the V07 has fully programmable configuration with no jumpers or straps, inventory is reduced to a single configuration. With the addition of SAM™-I/O modules, the system could consist entirely of identical foundation hardware and operating systems. Because the V07 lends itself to pre-configuration, it is ideal for the production environment.

The V07 uses a 68020 processor and a 68881 co-processor for its full 32-bit high-speed operation. It has Z8536 programmable configuration controller and 68155 bus interrupt manager. The baseboard has two serial ports configurable to RS232, RS422 and RS485.

Operating systems available with the V07 include PDOS, PSOS, VRTX and OS-9. Additional software includes a full-featured debugger and onboard diagnostics.

MOTOROLA INC.

Microprocessor Products Group
6501 William Cannon Drive West
Austin, Texas 78735-0598

EDITORIAL CONTACT:
Diane Falkenberg
512/928-6899

READER CONTACT:
Jane Bates
512/440-2030

INQUIRY RESPONSE:
Technical Info Center
P.O. Box 52073
Phoenix, AZ 85072

MOTOROLA ANNOUNCES 32-BIT IEEE 754-1985 COMPLIANT FLOATING POINT DIGITAL SIGNAL PROCESSORS

While fully supporting the IEEE 754-1985 binary floating point standard, the DSP96001 general purpose digital signal processor has a peak performance of 40 million floating point operations per second (MFLOPS) using a 26.7 MHz oscillator

Motorola Inc., Austin, Texas, March 30, 1988.......Motorola's Digital Signal Processor Operation, based in Austin, Texas, producer of the HYPERformance (TM) DSP56000 family of 24-bit digital signal processors, is announcing plans to launch an extension of its existing family. The DSP96001 will be the first available offering in the new DSP96000 Family equipped with floating point.

The DSP96001 is a 32-bit x 32-bit floating point digital signal processor that has 96-bit accumulators. The DSP96001 is in total conformance with the IEEE 754-1985 binary floating point standard. Software for the DSP96001 is both upward and downward compatible with the software for the DSP56000 family of fixed point digital signal processors. There are 512 words of on-chip full speed program RAM (PRAM), two 512-word data RAMS, two pre-programmed data ROMs, special on-chip bootstrap hardware for efficient program loading into the PRAM, an on-chip debug circuit to allow access to internal resources in support of the On-Chip Emulation, OnCE(TM), and two full DMA channels.

According to Bryant Wilder, Motorola's Manager for the digital signal processor operation, "the in-circuit emulation characteristics of board level products has been carried one step further. Motorola has brought that degree of debugging capability on-chip with the OnCE (TM) feature of the DSP96000 Family."

There are three 32-bit execution units operating in parallel within the CPU. These are the Data ALU, the Address Generation Unit, and the Program Controller. Having an MCU-like architecture with on-chip peripherals, program and data memory, and expansion ports, the DSP96001 offers substantial flexibility to the system designer for a variety of digital applications. Its programming model and instruction set are similar to that of a microprocessor, making the generation of efficient, compact code, straightforward for the programmer.

Operating at a peak performance rate of 40 MFLOPS, the DSP96001 is an ideal candidate to support a wide variety of real time DSP applications that demand the accuracy of an IEEE conformant floating point unit. Some of the more familiar applications would include high-speed controllers, digital audio systems, numeric processing, image and speech processing, spectral analysis, instrumentation, medical environments, and navigational systems.

Features that make the throughput possible for these application areas include the following:

Speed -- In less than 2 milliseconds, a 1024 point complex floating point Fast Fourier Transform (FFT) can be executed by the DSP96001 operating at a 13.33 million instructions per second (MIPS) rate.

Precision -- In single precision (SP), there is full IEEE 754-1985 floating point conformance. Full 32-bit buses and ALU support the SP operations by performing all floating point operations in single extended precision (SEP) arithmetic. This allows the SP error bounds to be met after only one of the four IEEE rounding operations have been executed - round to +infinity, - infinity, zero, and even. The general register file consists of ten 96-bit registers. IEEE double extended precision operations performed in software are supported. Enhancing precision one step further, the data ALU also supports 32-bit integer arithmetic including 32 x 32 integer multiplication with a full 64-bit product.

Parallelism -- The Data ALU, Address Generation Unit, and Program controller operate in parallel within the CPU so that an instruction prefetch, up to three floating point operations, two data moves, and two address pointer updates using one of three types of arithmetic (linear, modulo, or reverse carry) can be executed in a single instruction cycle. The 40 MFLOP peak performance is made possible as a result of this parallelism. Two on-chip DMA controllers operate unobtrusively in parallel with the CPU to assure there is an adequate supply of data available for processing.

Integration -- The DSP96001 is highly integrated. In addition to the three independent execution units and the two DMA controllers, there are six on-chip memories, three on-chip MCU style peripherals (serial communications interface, synchronous serial interface, and a 32-bit host interface), a clock generator and eight 32-bit wide buses (three address and five data). This makes the overall system high performance, low power, and compact -- an excellent cost performance value.

DSP96001 Compatibility -- The DSP96001 has the same basic architecture as the DSP56000 Family. Its instruction set is a superset of the DSP56000 Family instruction set. This compatibility means the hardware and software development tools are nearly identical for all Motorola DSPs, and therefore, investment in existing software is preserved. Applications can be easily developed on one processor and migrated across the Motorola DSP product line to meet various cost and performance objectives.

ATARI ST1040

Geza Holzhaker.
Apdo.Correos # 393.
Merida 5101. VENEZUELA.

Dear Mr. Williams:

I've just read the "ATARI call" in the Jan.88 issue.
I'd like very much if the 68MJ would include a regular section for the Atari computer.
Until today I've bought two ST1040's (Monochr. res.] 640x400). When the ST1040 came out, it had the worst Basic I've seen in my life. But now there is available GFA_Basic, it's structured and very fast, it has graphic capabilities and easy access to 68000 calls. There is also a useful assembler/disassembler (Assempro) to help writing 68000 codes, PC Board design, Publishing Partner and more.
I think the ST1040 is not expensive taking into account the graphic capabilities that other machines are lacking. The keyboard is not comfortable, but with all the software appearing the ST1040 is definitively not a toy.
I wish I could buy OS9 for the Atari some day, I am sure the Basic09 and "C" from Microware are better than others, but I don't know if they support graphics. I hope SK*DOS would be released too.

I've already installed a 5" (80 tracks) drive as drive "B" instead of another 3" drive. It works perfectly, even without slowing down the stepping rate of the drive (Teac-55F1. I feel 5" disks more reliable than 3" disks. I installed the 5" drive without having the schematics of the ST1040. I'd appreciate very (but very) much if somebody could send me a copy (and related info) of it to me. It is impossible to get them here in Venezuela.

At the present time we are developing intelligent Electrocardiographic and Electroencephalographic systems with the ST1040.

I hope the "ATARI" section will be a regular issue in the 68MJ.

Sincerely yours,

[illegible]

WINDRUSH
Micro Systems Ltd.

Worstead Laboratories (Reg. Office)
North Walsham, Norfolk NR28 9SA
Tel (0692) 404086
Telex 975548 WMICRO G
MAPCON approved consultants

MC68020 WORK STATION SUPPORTS STE & G64 BUSES

Windrush Micro Systems Limited are pleased to announce the immediate availability of Ωmega workstations with integral G64 or STE bus expansion facilities.

The new versions of the Ωmega enable users to take advantage of the wide range of off-the-shelf I/O modules available for these popular buses. Ten bus positions are available. The backplanes can be set at the normal depth which facilitates front access or the backplane can be set back so that all cards are hidden behind a hinged panel.

The standard range of Ωmega I/O cards can be used in conjunction with the bus expansion facilities. These include: A colour graphics adaptor which provides a 768 x 576 x 4 bits/pixel resolution. Up to sixteen colours from a pallatte of 4096 colours may be displayed simultaneously. The adaptor incoporates 512K of dedicated memory and the Hitachi HD63484 Advanced CRT controller. A nine port RS-232 serial port expansion card is also available as is a 3 Mb ROM expansion card.

The top of the line Ωmega workstation, costing £4950 incorporates a 16 MHz MC68020 processor and MC68881 math co-processor as standard and includes five RS-232 ports, a 40 Mb Winchester hard disc with a seek time of less than 30 mS, a 1 Mb 3.5 inch floppy disk, a 150 Mb 1/4" tape streamer, 2 megabytes of zero wait-state, non-volatile Static RAM. A parallel printer port, a clock calendar and OS-9/68K professional are also included. The G64 or STE expansion facilites add £495.

Other Ωmega configurations are available with prices starting at £1595 for the single disk 512K model with a 12 MHz MC68020 and OS-9/68K professional.

For further information contact Bill Dickinson at (0692) 404086

Classifieds As Submitted - No Guarantees

MUSTANG-020 16Mhz with 68881. OS9 Professional Package & C $3000. Call Tom (615)842-4600.

AT&T 7300 UNIX PC, UNIX V OS, 1MB Memory, 20 MB Hard Disk, 5" Drive, Internal Modem, Mouse. Best Offer Gets It.
S+System with Cabinet, 20 Meg Hard Disk & 8" Disk Drive with DMAF3 Controller Board. 1-X12 Terminal $4800.

DAISY WHEEL PRINTERS
Qume Sprint 9 - $900
Qume Sprint 5 - $800.

HARD DISK 10 Megabyte Drive - Seagate Model #412 $275.
3-Dual 8" drive enclosure with power supply. New in box. $125 each.
5-Siemens 8" Disk Drive, $100 each.

Tano Outpost II, 56K, 2 5" DSDD Drives, FLEX, MUMPS, $495.
QUME QVT-102 terminal, likenew, amber screen $250. or best offer.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09CPU Card- $900 complete.
Tom (615) 842-4600 M-F 9AM to 5PM EST

•••

For Sale: 6809 Microprocessor; 80 megabyte hard drive; floppy disk drive; tape backup; 8212 terminals; printer; uniflex operating system. Cabinetry and software available. $15,500.00 Contact Cal Mueller, Boelter Industries, Inc. Phone (507) 452-2315

APPLE MACINTOSH™ USERS

Save over a $1,000.00 on PostScript Laser Printers!
Faster - Finer Quality than the original Apple LaserWriter!
New & Demos
Cartridges-new-rebuilts
-colors-

In Chattanooga Call:
615 842-4600
QMS-Authorized

Data-Comp Division
"A Decade of Quality Service"
Systems World-Wide
Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4801 - Telex 510 600-6630 Hixson, Tn 37343

ELEKTRA™ Computer Products

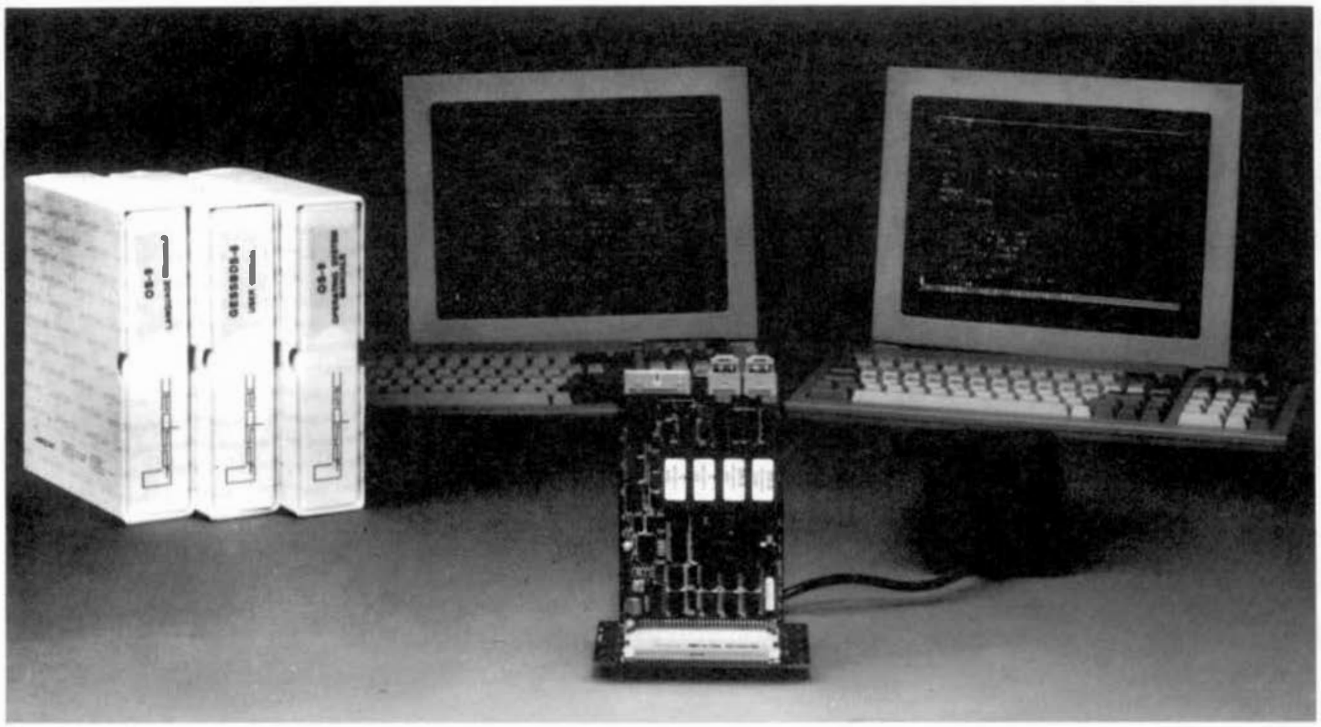
Complete line of SS-50 Systems and Components as well as MC680xx based Single Board Computers and Systems

In addition, **AAA Chicago Computer Center** is a dealer for, has inventory of items produced by, or services products produced by Computer Excellence, Computer System Center, Computer Systems Consultants Inc., GIMIX, Helix, Microware, Optimal Technology, Processor Technology, Smoke Signal Broadcasting, Southwest Technical Products, Sphere, Stylo Software Inc., Technical Systems Consultants, and Starkits.

Whether you are planning to add to an existing system or purchase a new system, be sure you phone **AAA Chicago Computer Center** first. Technical consultation is available weekdays 4 P.M. to 6 P.M. C.S.T. or leave a message and we'll call you back at your convenience.

AAA Chicago Computer Center
120 Chestnut Lane — Wheeling, IL 60090

(312) 459-0450 (till Nov. 1988) (708) 459-0450 (after Nov. 1988)



GESPAC's Multi-User Single Board System Will Get You Started With C and OS-9 for Only $1495

Introducing the GESSBDS-6. A revolutionary single board multi-user system with 512K of ROM resident software. The self-contained unit lets you program directly in C or 68000 assembly language under the powerful UNIX-like, OS-9 real-time operating system.

No need for external disk drives or RAM. The GESSBDS-6 provides a 128K non-volatile RAM disk for storing your source and object files. Detailed user documentation lets you begin writing your first program only minutes after unpacking the system.

The GESSBDS-6 is ready to grow with your needs. You can add floppy and hard disk storage, and up to 8 Mbytes of external memory through the board's G-64 bus interface. You can also expand your system with any of over 150 I/O modules manufactured by GESPAC.

The GESSBDS-6 is ideal for the development of small real-time imbedded control systems. You can also use the system to familiarize yourself with C, 68000 assembly, real-time multitasking programming technique and bus system architectures.

Ordering Information

GESSBDS-6	Single Board OS-9 Development System	$1495
POWR-1	Optional 20W external power supply	$110

Software included

OS-9 V2.1	Operating system and 27 utilities
c68	C language compiler
debug	Symbolic debugger
l68	Relocatable linker
r68	68000 macro assembler
uMACS	Screen oriented text editor

Hardware specifications

- 8 MHz 16/32-bit 68000 CPU
- 512K of EPROM
- 256K of CMOS Static RAM
- Three 16-bit counters/timers
- Real-Time-Clock/Calendar
- Two 8-bit bidirectional TTL I/O port
- Two RS-232 serial ports
- 8 Mbytes of external memory addressing
- +5V @ 1.9A, +/-12V @ 50mA

For More Information or to Place Order Call 1-800-4-GESPAC or (602)-962-5559

Gespac

50 West Hoover Ave.
Mesa, AZ 85210
Phone (602) 962-5559
Fax (602) 962-5750

NEW!

OmegaSoft Pascal for the 68020/68881

P20K is a Pascal package that will generate code for all of the 68000 series processors, including the 68881 coprocessor. P20K will run on any 68000 series computer running the OS-9/68000 (Microware) or PDOS (Eyring Research) operating systems with 512K or more free memory.

The base package (P20K-B) includes the Compiler, Relocatable Macro Assembler, Linking Loader, Screen Editor, Pascal Shell, Linkage Creator, Host Debugger, Configuration manager, Installation program, and Patch utility. A new feature in this compiler is the ability to either link in the parts of the runtime needed by the program, or to use trap handlers for runtime access to share the runtime library between programs. Complete operating system interface is also included using pascal procedures and functions. The host debugger allows debugging at both the Pascal and assembly language levels of programs that run on the host operating system. Price for the base package is $575.

The runtime source code option (P20K-R) is available for $100 and includes source code for the operating system interface routines as well as pascal runtime.

The Utility source option (P20K-S) is available for $275 and includes source code for the Screen Editor, Pascal Shell, Host Debugger, Patch utility, and Configuration manager.

The Target debugger option (P20K-T) is $225 and includes object and source code. This program allows Pascal level and assembly level debugging in a system without operating system, by using a serial link connected to the host computer.

Prices do not include shipping charges. Master-Card and Visa accepted. OmegaSoft is a registered trademark of Certified Software Corporation.

Gespac SA, 3, Chemin des Aulx, CH-1228, Geneva/Plan-les-Ouates, Switzerland
TEL 022-713400, TLX 429989

Elsoft AG, Zelgweg 12, CH-5405 Baden-Dättwil, Switzerland TEL 056-833377, TLX 828275

RCS Microsystems Ltd, 141 Uxbridge Road, Hampton Hill, Middlesex, England
TEL 01-9792204, TLX 8951470

Byte Studio Borken, Butenwall 14, D-4280 Borken, West Germany.
TEL 02861-2147, TLX 813343

Eltec Elektronik GmbH, Galileo-Galilei-Straße, 6500 Mainz 42, Postfach 65, West Germany
TEL 06131-50031, TLX 4187273

PEP Elektronik Systeme GmbH, Am Klosterwald 4 D-8950 Kaufbeuren, West Germany
TEL 08341-8974, TLX 541233

CERTIFIED
SOFTWARE
CORPORATION

616 CAMINO CABALLO, NIPOMO, CA 93444 USA
TEL (805) 929-1395 TELEX 467013
FAX (805) 929-1395 (MID-8AM)

FLEX™ MS-DOS™

Transfer Utilities

For 68XXX and CoCo* OS-9™ Systems

Now READ - WRITE - DIR - DUMP - EXPLORE
FLEX & MS-DOS Disk

These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks.

*CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.

CoCo Version: $69.95 68XXX Version $99.95

S.E. Media ←→ PO Box 849, Hixson,TN 37343
615 842-6809 MC/Visa

SUPERIOR SOFTWARE
FOR YOUR
6809

★ SK•DOS® Disk Operating System	$75.00
★ Configuration Manual	$50.00
★ HUMBUG® Monitor	$50.00
★ MICROBUG Monitor	$30.00
★ SPELL 'N FIX Spelling Checker	$89.29
★ STAR-DOS for the Coco	$34.50
★ CHECK 'N TAX	$50.00

AND 68000

★ SK•DOS® Operating System	$140.00
★ HUMBUG® Monitor	$ 50.00
★ SPELL 'N FIX Spelling Checker (Coming)	

STAR-K
SOFTWARE SYSTEMS CORPORATION
BOX 209 • MT. KISCO, N.Y. 10549 • 914/241-0287

TIRED OF SLOW COMPUTERS ??
PROGRAMS SEEM TO TAKE FOREVER ??
DETROIT DATACOMM OFFERS A SOLUTION !!

A hardware emulator executing 68000 object code !!

Using a 60Mhz clock with a floating point multiplier frok Bipolar Intergrated Technology the system is capable of up to 5 MIPS.
- By running EXISTING software there is NO need to redo or convert programs
- SWAP AND GO remove your existing CPU, plug in the hardware emulator and watch your program run in record time
- Performs hardware square roots a real time SAVER !!

Available July 1988

Detroit Datacom Inc.
1404 West 14 Mile Road
Madison Heights, MI. 48071
1-(313)-524-2868

SOFTWARE FOR 680x AND MSDOS

SUPER SLEUTH DISASSEMBLERS
EACH $99-FLEX $101-OS9 $100-UNIFLEX
OBJECT-ONLY versions: EACH $50-FLEX,OS9,COCO
interactively generate source on disk with labels, include xref, binary editing
specify 6800,1,2,3,5,8,9/6502 version or Z80/8080,5 version
OS9 version also processes FLEX format object file under OS9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/8080,5) only
68010 disassembler $100-FLEX,OS9,UNIFLEX,MSDOS,UNIX,SKDOS

CROSS-ASSEMBLERS WITH MACRO CAPABILITIES
EACH $50-FLEX,OS9,UNIFLEX,MSDOS,UNIX,SKDOS 3/$100 ALL/$200
specify: 180x,6502,6801/11,6804,6805,6809,Z8,Z80,8048,8051,8085,68010,32000
modular cross assemblers in C, with load/unload utilities
sources for additional $50 each, $100 for 3, $300 for all

DEBUGGING SIMULATORS FOR POPULAR 8-BIT MICROPROCESSORS
EACH $75-FLEX $100-OS9 $80-UNIFLEX
OBJECT-ONLY versions: EACH $50-COCO FLEX,COCO OS9
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (14)6805, 6502, 6809 OS9, Z80 FLEX

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809
6502 to 6809 $75-FLEX $85-OS9 $80-UNIFLEX
6800/1 to 6809 & 6809 to position ind. $50-FLEX $75-OS9 $60-UNIFLEX

FULL-SCREEN XBASIC PROGRAMS with cursor control
AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS

DISPLAY GENERATOR/DOCUMENTOR	$50 w/source, $25 without
MAILING LIST SYSTEM	$100 w/source, $50 without
INVENTOR WITH MRP	$100 w/source, $50 without
TABULA RASA SPREADSHEET	$100 w/source, $50 without

DISK AND XBASIC UTILITY PROGRAM LIBRARY
$50-FLEX $30-UNIFLEX/MSDOS
edit disk sectors, sort directory, maintain master catalog, do disk sorts,
resequence some or all of BASIC program, xref BASIC program, etc.
non-FLEX versions include sort and resequence only

CMODEM TELECOMMUNICATIONS PROGRAM
$100-FLEX,OS9,UNIFLEX,MS-DOS,UNIX,SKDOS
OBJECT-ONLY versions: EACH $50
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc.
for COCO and non-COCO; drives internal COCO modem port up to 2400 Baud

DISKETTES & SERVICES

5.25" DISKETTES
EACH 10-PACK $7.50-SSSD/SSDD/DSDD
American made, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

ADDITIONAL SERVICES FOR THE COMPUTING COMMUNITY

CUSTOMIZED PROGRAMMING
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

CONTRACT PROGRAMMING
we will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 680x0, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

CONSULTING
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone 404-483-4570 or 1717

We take orders at any time, but plan long discussions after 6, if possible.

Contact us about catalog, dealer, discounts, and services.
Most programs in source: give computer, OS, disk size.
25% off multiple purchases of same program on one order.
VISA and MASTER CARD accepted; US funds only, please.
Add GA sales tax (if in GA) and 5% shipping.
(UNIFLEX tm Technical Systems Consultants; OS9 Microware; COCO Tandy; MSDOS Microsoft; SKDOS Star-K Software)

K-BASIC™

The Only 6809 BASIC to Binary Compiler for OS-9
FLEX or SK*DOS
Even runs on the 68XXX SK*DOS Systems*

Hundreds Sold at Suggested Retail:
~~$199.00~~

• 6809 - OS-9™ users can now transfer their FLEX™ Extended BASIC (XBASIC) source files to OS-9, compile with the OS-9 version and run them as any other OS-9 binary "CMD" program. Much faster than BASIC programs.

• 6809 - FLEX users can compiler their BASIC source files to a regular FLEX *.CMD" file. Much faster execution.

• 68XXX - SK*DOS™ users running on 68XXX systems (such as the Mustang-08/A) can continue to execute their 6809 FLEX BASIC and compiled programs while getting things ported over to the 68XXX. SK*DOS allows 6809 programs to run in emulation mode. This is the only system we know of that will run both 6809 & 68XXX binary files.

K-BASIC is a true compiler. Compiling BASIC 6809 programs to binary command type programs. The savings in RAM needed and the increased speed of binary execution makes this a must for the serious user. And the price is now RIGHT!

Don't get caught up in the *"Learn a New Language"* syndrome - Write Your Program in BASIC, Debug It in BASIC and Then Compile It to a .CMD Binary File.

For a LIMITED time save over 65%... This sale will not be repeated after it's over! *

SALE SPECIAL:
$69.95

SPECIAL

Thank-You-Sale

Only From:

S.E. Media™

CPI

5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

A Division of Computer Publishing Inc.
Over 1,200 Titles - 6800-6809-68000

* K-BASIC will run under 68XXX SK*DOS in emulation mode for the 6809.
Price subject to change without notice.

Clearbrook Software Group (604)853-9118

CSG IMS is *THE* full featured relational database manager for OS9/OSK. The comprehensive structured application language and B+Tree index structures make **CSG IMS** the ideal tool for file-intensive applications.

CSG IMS for CoCo2/3 OS9 L1/2 (single user)	$169.95
CSG IMS for OS9 L2 or 68000(multi user)	$495.00
CSG IMS demo with manual	$30

MSF - MSDos File Manager for CoCo 3/OS9 Level 2
allows you to use MSDos disks directly under OS9.
Requires CoCo 3, OS9 L2, SDISK3 driver $45.00

SERINA - System Mode Debugger for OS9 L2
allows you to trace execution of any system module, set break points, assemble and disassemble code and examine and change memory.
Requires CoCo3 or Gimix II, OS9 L2 & 80 col. terminal $139.00

ERINA - Symbolic User Mode Debugger for OS9
lets you find bugs by displaying the machine state and instuctions being executed. Set break points, change memory, assemble and disassemble code.
Requires 80 column display, OS9 L1/2 $69.00

Shipping: N. America - $5, Overseas - $10
Clearbrook Software Group P.O. Box 8000-499, Sumas, WA 98295
OS9 is a trademark of Microware Systems Corp., MSDos is a trademark of Microsoft Corp.

SPECIAL

ATARI™
&
OS-9™

NOW!
If you have either the
Atari 520 or 1040 -
you can take
advantage of the
"bargain of a lifetime"
OS-9 68K and BASIC
all for the low, low price of:

$150.00

Call or Write
S.E. Media
5900 Cassandra Smith Rd.
Hixson, TN 37343
615 842-4601

ATARI & AMIGA CALL

As most of you know, we are very sensitive to your wishes, as concerns the contents of these pages. One of the things that many of you have repeatedly written or called about is coverage for the **Atari & Amiga™** series of 68000 computers.

Actually we haven't been too keen on those systems due to a lack of serious software. They were mainly expensive "game-toy" systems. However, recently we are seeing more and more honest-to-goodness serious software for the Atari & Amiga machines. That makes a difference. I feel that we are ready to start some serious looking into a section for the Atari & Amiga computers. Especially so since OS-9 is now running on the Atari (review copy on the way for evaluation and report to you) and rumored for the Amiga. Many of you are doing all kinds of interesting things on these systems. By sharing we all benefit.

This I must stress - *Input from you on the Atari & Amiga.* As most of you are aware, we are a "contributor supported" magazine. That means that YOU have to do your part. Which is the way it has been for over 10 years. We need articles, technical, reviews of hardware and software, programming (all languages) and the many other facets of support that we have pursued for these many years. Also I will need several to volunteer to do regular columns on the Atari & Amiga systems. Without constant input we can't make it fly! So, if you do your part, we certainly will do ours. How about it, drop me a line or give me a phone call and I will get additional information right back to you. We need your input and support if this is to succeed!

DMW

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles **OS9 USER NOTES**

Information for the BEGINNER to the PRO, Regular or CoCo OS9

Using OS9
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

Program interfacing to OS9
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include
No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled Operating Programs. The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing your OWN more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC

NOTE: .C1,.C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail
or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
*** All Currency in U.S. Dollars**

Continually Updated In 68 Micro Journal Monthly

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343

VISA

(615) 842-4601
Telex 5106006630

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware and Motorola
"68' Micro Journal is a trademark of Computer Publishing Inc.

!!! Subscribe Now !!!
68 MICRO JOURNAL

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Mastercard ☐ VISA ☐

Card #____________________ Exp. Date________

For 1 Year ____ 2 Years ____ 3 Years ____

Enclosed: $__________

Name ____________________________

Street ____________________________

City ________________ **State** ____ **Zip** ______

Country ____________________________

My Computer Is: ______________

Subscription Rates

U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50

•Foreign Surface: Add $12.00 per Year to USA Price.

•Foreign Airmail: Add $48.00 per Year to USA Price.

•Canada & Mexico: Add $9.50 per Year to USA Price.

•U.S. Currency Cash or Check Drawn on a USA Bank !

68 Micro Journal
5900 Cassandra Smith Rd.
POB 849
Hixson, TN 37343

Telephone 615 842-4600
Telex 510 600-6630

Reader Service Disks

Disk- 1	Fileson, Minical, Minicopy,Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.
Disk- 2	Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn,**Lifetime.
Disk- 3	Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.
Disk- 4	Mailing Program, *Finddat, *Change, *Testdisk.
Disk- 5	*DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
Disk- 6	**Purchase Order, Index (Disk file indx).
Disk- 7	Linking Loader, Rload, Harkness.
Disk- 8	Crtest, Lanpher (May 82).
Disk- 9	Datecopy, Diskfix9 (Aug 82).
Disk-10	Home Accounting (July 82).
Disk-11	Dissembler (June 84).
Disk-12	Modem68 (May 84).
Disk-13	*Initmf68, Testmf68,*Cleanup,*Dskalign,Help,Date.Txt.
Disk-14	*Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib.
Disk-15	Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo).
Disk-16	Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
Disk-17	Match Utility, RATBAS, A Basic Preprocessor.
Disk-18	Parse.Mod, Size.Cmd (Sept. 85 Armstrong),CMDCODE, CMD.Txt (Sept. 85 Spray).
Disk-19	Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
Disk-20	UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C.
Disk-21	Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
Disk-22	Read CPM & Non-FLEX Disks. Fraser May 1984.
Disk-23	ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven, Language Recognition Utility, Anderson March 1986.
Disk-24	68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
Disk-25	KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
Disk-26	Compacta UniBoard review, code & diagram. Burlison March '86.
Disk-27	ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT.
Disk-28	CT-82 Emulator, bit mapped.
Disk-29	**Star Trek
Disk-30	Simple Winchester, Dec.'86 Green.
Disk-31	*** Read/Write MS/PC-DOS (SK*DOS)
Disk-32	Heir-UNIX Type upgrade - 68MJ 2/87
Disk-33	Build the GT-4 Terminal - 68MJ 11/87 Condon.
Disk-34	FLEX 6809 Diagnostics, Disk Drive Test, ROM Test, RAM Test - 68MJ 4/88Korpi.

NOTE:

This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

8" disk $19.50
5" disk $16.95

Shipping & Handling -U.S.A. Add: - $3.50
Overseas add: $4.50 Surface - $7.00 Airmail

68 MICRO JOURNAL
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4600 - Telex 510 600-6630

PT-68000 SINGLE BOARD COMPUTER

The PT68K2 is Available in a Variety of Formats
From Basic Kits to Completely Assembled Systems

BASIC KIT (8 MHZ) - Board, 68000, HUMBUG MONITOR + BASIC in ROM, 4K STATIC RAM, 2 SERIAL PORTS, all Components $200

PACKAGE DEAL - Complete Kit with Board 68000 10 MHZ, SK*DOS, 512K RAM, and all Necessary Parts $530

ASSEMBLED BOARD (12 MHZ) Completely Tested, 1024K RAM, FLOPPY CONTROLLER, PIA, SK*DOS $849

ASSEMBLED SYSTEM - 10 MHZ BOARD, CABINET POWER SUPPLY, MONITOR + KEYBOARD, 80 TRACK FLOPPY DRIVE, CABLES $1299
For A 20 MEG DRIVE, CONTROLLER and CABLES Add $345

PROFESSIONAL OS9 $500

FEATURES

- MC68000 Processor, 8 MHZ Clock (optional 10,12.5 MHZ)
- 512K or 1024K of DRAM (no wait states)
- 4K of SPAM (6116)
- 32K,64K or 128K of EPROM
- Four RS-232 Serial Ports
- Floppy disk controller will control up to four 5 1/4", 40 or 80 track.
- Clock with on-board battery.
- 2 - 8 bit Parallel Ports
- Board can be mounted in an IBM type PC/XT cabinet and has a power connector to match the IBM type power supply.
- Expansion ports - 6 IBM PC/XT compatible I/O ports. The HUMBUG™ monitor supports monochrome and/or color adaptor cards and Western Digital winchester interface cards.

PERIPHERAL TECHNOLOGY
1480 Terrell Mill Rd., Suite 870
Marietta, Georgia 30067
404/984-0742
VISA/MASTERCARD/CHECK/C.O.D.

Send For Catalogue
For Complete Information On All Products

™SK*DOS is a Trademark of STAR-K SOFTWARE SYSTEMS CORP.
™OS9 is a Trademark of Microware

DATA-COMP SPECIAL

Heavy Duty Power Supplies

For A limited time our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units. Note that these prices are less than 1/4 the normal price for these high quality units.

Make: Boschert
Size: 10.5 x 5 x 2.5 inches
Including heavy mounting bracket and heatsink

Rating: in 110/220 volts ac (strap change) Out: 130 watts

Output: +5v - 10 amps
+12v - 4.0 amps
+12v - 2.0 amps
-12v - 0.5 amps

Mating Connector: Terminal strip
Load Reaction: Automatic short circuit recovery
SPECIAL: $59.95 each
2 or more $49.95 each
Add: $7.50 each S/H

Make: Boschert
Size: 10.75 x 6.2 x 2.25 inches

Rating: 110/220 ac (strap change) Out: 81 watts

Outputs: +5v - 8.0 amps
+12v - 2.4 amps
+12v - 2.4 amps
+12v - 2.1 amps
-12v - 0.4 amps

Mating Connectors: Molex
Load Reaction: Automatic short circuit recovery
SPECIAL: $49.95 each
2 or more $39.95 each
Add: $7.50 S/H each

5900 Cassandra Smith Rd., Hixson, Tn. 37343 Telephone 615 842-4600 Telex 510 600-6630

GMX MICRO-20 and TWINGLE-20 PRICE LIST

All versions include 1 SAB Board

	MICRO-20 with 1MB RAM	MICRO-20 with 2MB RAM	TWINGLE-20 with 4MB RAM
12.5 MHz	1855.00	2155.00	3855.00
16.67 MHz	2185.00	2485.00	4185.00
20 MHz	2585.00	2885.00	4785.00

OPTIONAL PARTS AND ACCESSORIES

68881 12.5MHz Floating Point Coprocessor	$ 165.00
68881 16.67MHz Floating Point Coprocessor	$ 225.00
68881 20MHz Floating Point Coprocessor	$ 345.00
MOTOROLA 68020 USERS MANUAL	$ 18.00
MOTOROLA 68030 USERS MANUAL	$ 18.00
MOTOROLA 68881 USERS MANUAL	$ 18.00

SBC ACCESSORY PACKAGE (M20-AP) $1399.00

The package includes a PC-style cabinet with a custom backpanel, a 25 Megabyte (unformatted) hard disk and controller, a floppy disk drive, a 150 watt power supply, cooling fan, panel mounted reset and abort switches, and all necessary internal cabling. (For use with SAB—9D serial connectors only.)

2nd 5"80 FLOPPY & CABLES FOR M20-AP, ADD	$ 250.00
SECOND 25MB HARD DISK & CABLES, ADD	$ 780.00
TO SUBSTITUTE 50MB HD FOR 25MB HD, ADD	$ 290.00
TO SUBSTITUTE 80MB HD FOR 25MB HD, ADD	$1500.00
TO SUBSTITUTE 155MB FOR 25MB HD, ADD	$2100.00
60MB TEAC STREAMER WITH ONE TAPE	$ 690.00
PKG. OF 5 TEAC TAPES	$ 112.50
CUSTOM BACK PANEL PLATE (BPP-PC)	$ 44.00

I/O EXPANSION BOARDS

16 PORT SERIAL BOARD ONLY (SBC-16S) .... $ 335.00

The SBC-16S extends the I/O capabilities of the GMX Micro-20 68020 Single-board Computer by adding sixteen asynchronous serial I/O ports. By using two SBC-16S boards, a total of thirty-six serial ports are possible.

RS232 ADAPTER (SAB-25, SAB-9D or SAB-8M) $165.00

The board provides level-shifting between TTL level and standard RS-232 signal levels for up to 4 serial I/O ports.

60 LINE PARALLEL I/O BOARD (SBC-60P) $398.00

The GMX SBC-60P uses three 68230 Parallel Interface/Timers (PI/Ts) to provide up to forty-eight parallel I/O lines. The I/O lines are buffered in six groups of eight lines each, with separate buffer direction control for each group. Buffer direction can be fixed by hardware jumpers, or can be software programmable for bidirectional applications.

PROTOTYPING BOARD (SBC-WW) $75.00

The SBC-WW provides a means of developing and testing custom I/O interface designs for the GMX Micro-20 68020 Single-board Computer. The board provides areas for both DIP (Dual Inline Package) and PGA (Pin Grid Array) devices, and a pre-wired memory area for up to 512K bytes of dynamic RAM.

I/O BUS ADAPTER (SBC-BA) $195.00

The SBC-BA provides an interface between the GMX Micro-20 68020 Single-board Computer and the Motorola Input/Output Channel (I/O bus). With the I/O bus, up to sixteen off-the-shelf or custom peripheral devices (I/O modules) can be connected to the GMX Micro-20.

ARCNET LAN board w/o Software (SBC-AN) $475.00

The SBC-AN provides an interface between the GMX Micro-20 68020 Single-board Computer and the ARCNET modified token-passing Local Area Network (LAN) originally developed by Datapoint Corp. The ARCNET is a baseband network with a data transmission rate of 2.5 Megabits/second. The standard transmission media is a single 93 ohm RG-62/U coaxial cable. Fiber optic versions are available as an option.

OS9 LAN Software Drivers for SBC-AN 120.00

GMX MICRO-20 SOFTWARE

020 BUG UPDATE — PROMS & MANUAL $150.00

THESE 68020 OPERATING SYSTEMS ARE PRICED WHEN PURCHASED WITH THE MICRO-20. PLEASE ADD $150.00 IF PURCHASED LATER FOR THE UPDATED PROMS AND MANUALS. ALL SHIPPED STANDARD ON 5¼" DISKS 3½" OPTIONAL IF SPECIFIED.

OS9/6 020 PROFESSIONAL PAK $850.00

Includes O.S., "C", uMACS EDITOR, ASSEMBLER, DEBUGGER, development utilities, 68881 support.

OS9/6 020 PERSONAL PAK .... $ 400.00

Personal OS-9 systems require a GMX Micro-20 development system running Professional OS-9/68020 for initial configuration.

BASIC (Included in PERSONAL PAK)	$ 200.00
C COMPILER (included in PROFESSIONAL PAK)	$ 750.00
PASCAL COMPILER	$ 500.00

UniFLEX (for Micro-20)	$ 400.00
UniFLEX WITH REAL-TIME ENHANCEMENTS	$ 800.00
UniFLEX VM (for TWINGLE-20)	$ 600.00
UniFLEX VM REAL-TIME ENHANCEMENTS	$1000.00

Other Software for UniFLEX

UniFLEX BASIC W/PRECOMPILER	$ 300.00
UniFLEX C COMPILER	$ 350.00
UniFLEX COBOL COMPILER	$ 750.00
UniFLEX SCREEN EDITOR	$ 150.00
UniFLEX TEXT PROCESSOR	$ 200.00
UniFLEX SORT/MERGE PACKAGE	$ 200.00
UniFLEX VSAM MODULE	$ 100.00
UniFLEX UTILITIES PACKAGE I	$ 200.00
UniFLEX PARTIAL SOURCE LICENSE	$1000.00

GMX EXCLUSIVE VERSIONS, CUSTOMIZED FOR THE MICRO-20, OF THE BELOW LANGUAGES AND SOFTWARE ARE ALSO AVAILABLE FROM GMX.

ABSOFT FORTRAN (UniFLEX)	$1500.00
SCULPTOR (specify UniFLEX or OS9)	$ 995.00
FORTH (OS9)	$ 595.00
DYNACALC (specify UniFLEX or OS9)	$ 300.00

GMX DOES NOT GUARANTEE PERFORMANCE OF ANY GMX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

ALL PRICES ARE F.O.B. CHICAGO IN U.S. FUNDS

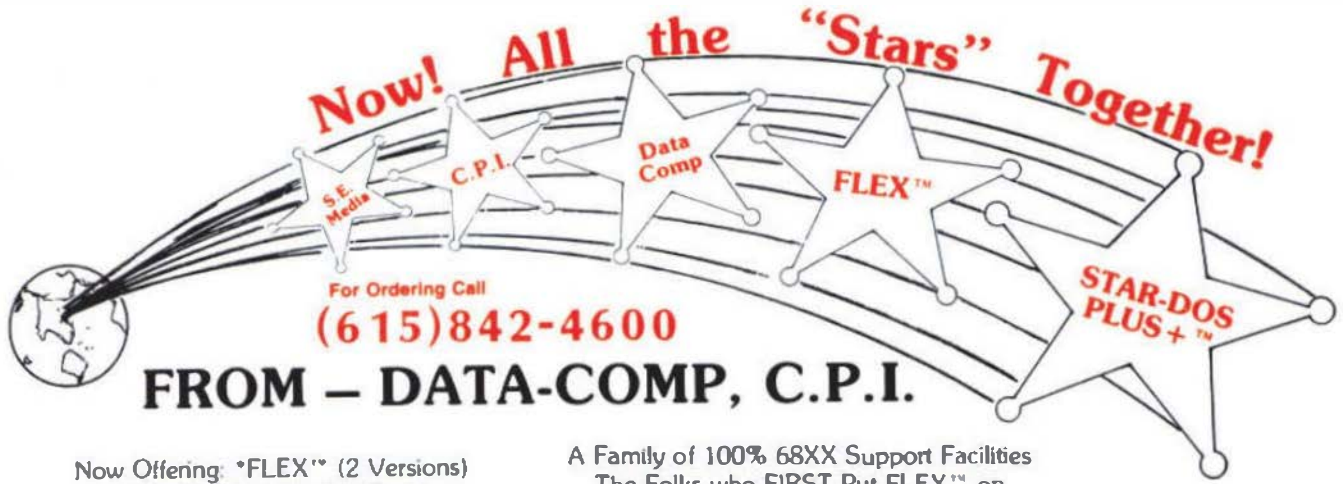
GMX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account number 73-32033.

CONTACT GMX FOR MORE INFORMATION ON THE ABOVE PRODUCTS

GMX STILL SELLS GIMIX SS0 BUS SYSTEMS, BOARDS & PARTS. CONTACT GMX FOR COMPLETE PRICE LIST.

GMX™ 1337 W. 37th Place, Chicago, IL 60609 (312) 927-5510 — TWX 910-221-4055 — FAX (312) 927-7352



For Ordering Call
(615)842-4600

FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.95

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities

+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY, CABINET, DISK DRIVE CABLE, J&M NEW DISK CONTROLLER JFD-CP WITH J-DOS, RS-DOS OPERATING SYSTEMS.	$469.95

* Specify What CONTROLLER You Want J&M, or RADIO SHACK

THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS	$129.95

Verbatim Diskettes

Single Sided Double Density	$ 24.00
Double Sided Double Density	$ 24.00

Controllers

J&M JFD-CP WITH J-DOS	$139.95
WITH J-DOS, RS-DOS	$159.95
RADIO SHACK 1.1	$134.95

RADIO SHACK Disk CONTROLLER 1.1	$134.95

Disk Drive Cables

Cable for One Drive	$ 19.95
Cable for Two Drives	$ 24.95

MISC

64K UPGRADE FOR C,D,E,F, AND COCO II	$ 29.95
RADIO SHACK BASIC 1.2	$ 24.95
RADIO SHACK DISK BASIC 1.1	$ 24.95
DISK DRIVE CABINET FOR A SINGLE DRIVE	$ 49.95
DISK DRIVE CABINET FOR TWO THINLINE DRIVES	$ 69.95

PRINTERS

EPSON LX-80	$289.95
EPSON MX-70	$125.95
EPSON MX-100	$495.95

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD	$ 89.95
8149 32K EXPAND TO 128K	$169.95
EPSON MX-RX-80 RIBBONS	$ 7.95
EPSON LX-80 RIBBONS	$ 5.95
TRACTOR UNITS FOR LX-80	$ 39.95

CABLES & OTHER INTERFACES
CALL FOR PRICING

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
Telex 5106006630

An Ace of a System in Spades! The New

MUSTANG-08/A™

Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the $500.00 OS-9 C compiler! This offer won't last forever!

NOT 128K, *NOT* 512K

FULL 768K No Wait RAM

The **MUSTANG-08™** system took every hand from all other 68008 sysems we tested, running OS-9 68K!

The **MUSTANG-08** includes OS9-68K™ and/or Peter Stark's SK*DOS™. SK*DOS is a single user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actually a 68XXX FLEX type system *(Not a TSC product.)*

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a speed *whiz on disk I/O*. Fact is: the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! See benchmarks!

SALE!

System includes OS-9 68K or SK*DOS - Your Choice

Specifications:

CPU	MC68008	12 Mhz
RAM	768K	256K Chips
	No Wait States	
PORTS	4 - RS232	MC68681 DUART
	2 - 8 bit Parallel	MC6821 PIA
CLOCK	MK48T02	Real Time Clock Bat. B/U
EPROM	16K, 32K or 64K	Selectable
FLOPPY	WD1772	5 1/4 Drives
HARD DISK	Interface Port	WD1002 Board

Now more serial ports - faster CPU Battery B/U - and $850.00 OS-9 Professional with C compiler included!

***$400.00** See Mustang-02 Ad - page 5 for trade-in details

Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

MUSTANG-08

L O O K

```
                              Seconds   32 bit    Register
                                        Integer   Long
Other    68008  8  Mhz  OS-9  68K...18.0...9.0
MUSTANG-08  10  Mhz  OS-9  68K....9.8...6.3
          Main()
              {
C Benchmark Loop      /* int i; */
                      register long i;
                      for (i=0; i < 999999; ++i);
              }
```

Now even faster! with 12 Mhz CPU

C Compile times: OS-9 68K Hard Disk	
MUSTANG-08 8 Mhz CPU	0 min - 32 sec
Other popular 68008 system	1 min - 05 sec
MUSTANG-020	0 min - 21 sec

♠ 25 Megabyte Hard Disk System

$2,398.90

Complete with PROFESSIONAL OS-9 includes the $500.00 C compiler, PC style cabinet, heavy duty power supply, 5" DDDS 80 track floppy, 25 MegByte Hard Disk - Ready to Run

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC   MUSTANG-08 is a trademark of CPI

Data-Comp Division

"A Decade of Quality Service"

Systems World-Wide

CPI Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

* Those with SWTPC hi-density FLEX 5" - Call for special info.